



António Nunes Jorge

Licenciado em Ciências da Engenharia Eletrotécnica e Computadores

Reserved Parking Validation

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e Computadores

Orientador: Rui Manuel Leitão Santos Tavares
PhD, Auxiliar Professor
Electrical Engineering Department
NOVA School of Science and Technology
Universidade NOVA de Lisboa

Júri:

Presidente: Rodolfo Alexandre Duarte Oliveira
PhD, Auxiliar Professor
FCT/UNL

Arguentes: José Manuel Matos Ribeiro da Fonseca
PhD, Auxiliar Professor
FCT/UNL

Vogais: Rui Manuel Leitão Santos Tavares
PhD, Auxiliar Professor
FCT/UNL

Setembro, 2019



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Reserved Parking Validation

Copyright © António Nunes Jorge, Faculty of Sciences and Technology, NOVA University Lisbon.

The Faculty of Sciences and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

*To my son and daughter, to my parents and to my wife.
In memory to my grandparents
We're never too old to fulfill our dreams*

Acknowledgements

I should start to thank professor Rui Tavares. After putting up with me for five years as a fellow student, 20 years later, he “inherited” the “almost impossible mission” of guiding and helping me building this dissertation, and to navigate this ship to safe waters.

His capacity of guidance, influencing and making me find the right path, but without making the decisions, letting me take that responsibility, is a very brilliant way to help and guide, mainly in a situation when we are more or less the same age, and I have a professional experience of almost 20 years.

His motivation capacity is of another league, even when I bumped in serious adversities, he was always there, with his always positive vision, with the glass half full perspective, at times when I though the glass was half empty (or with only a few drops).

I need to thank my wife for putting up with me and my bad temper for all these years, and for supporting me, in fulfilling my dreams. To my son and daughter, who during this time saw some of their activities and weekend escapes canceled, but I think that one day they will understand this effort, and that my decision to “go back to school” was also because of them and for them.

To my parents, thank you for all the investment and support, for all these years. They were the biggest victims, when in 1999, I decided to pursue a professional career, thinking I could do it at the same time I finished my studies. They were the wise ones ...

I also need to thank my colleagues and my employer, for all the given flexibility, patience, and support when I needed them.

Lastly, I need to thank my friends, who supported and helped me, sometimes with just a friendly word, or like Pedro Sanches, by going together with me to a Metal concert.

Abstract

A common situation that we can testify every day: fossil fuel cars occupying electric cars charge only places, and handy capped reserved places, occupied with cars without the proper authorization.

This is something that plagues our society, where the values and moral are forgotten, and our duties and rights are lost in the day-to-day life. There are more and more cars moving, every day, to the city center, where the lack of available parking, together with the lack of proper public transportation creates a chaotic situation. Also, the large proliferation of electric cars, that is not accompanied by a proportional availability of electric chargers, raises issues, where these cars' drivers are not allowed to charge their vehicles, most of the times, because they are being used as abusive parking.

This dissertation has the goal to identify and propose a universal solution, with low implementation and maintenance costs, that allows a fast and unambiguous validation of authorization of a user, for parking in a reserved parking space.

Keywords: parking, validation, authorization, reserved, cost, IoT, smart parking, smart cities, cloud

Resumo

Uma visão comum nos dias de hoje: veículos movidos a combustíveis fósseis a ocupar lugares reservados ao carregamento de veículos elétricos, e a ocupação indevida de lugares reservados ao estacionamento de deficientes, por veículos sem a devida autorização.

Este é um mal que afeta a nossa sociedade, onde cada vez mais a moral e valores e a capacidade de respeitar os nossos deveres e os direitos dos outros, são esquecidos no dia-a-dia. Também a cada vez maior proliferação de veículos particulares no centro das cidades, onde o estacionamento não pago é escasso, o que combinado com uma inadequada oferta de transportes públicos, é uma receita para o caos. Por outro lado, o não acompanhamento do aumento da venda de veículos elétricos, com a disponibilização de um número suficiente de carregadores, provoca situações, em que os utilizadores destes veículos, se vêm incapacitados de carregar os mesmos, devido ao facto dos lugares destinados a este fim, serem inúmeras vezes utilizados em estacionamento abusivo.

Esta dissertação tem como objetivo identificar e propor uma solução universal, com custos de implementação e manutenção reduzidos, e que permita uma rápida validação, sem ambiguidade, da autorização de estacionamento de um dado utilizador, num lugar reservado.

Palavras-chave: estacionamento, validação, autorização, reservado, custo, IoT, estacionamento inteligente, cidades inteligentes, cloud

Contents

TABLES LIST	XV
FIGURES LIST	XVII
ACRONYMS LIST	XXI
CHAPTER 1 – INTRODUCTION.....	1
1.1 - CONTEXT	1
1.2 – MOTIVATION AND OBJECTIVES.....	2
1.3 - DOCUMENT STRUCTURE.....	2
CHAPTER 2 – STATE OF THE ART	5
2.1 – SMART CITIES AND THE INTERNET OF THINGS	5
2.2 – PROPOSALS PRESENTATION	7
2.3 – ADVANTAGES AND DISADVANTAGES RESUME.....	20
2.4 – CONCLUSIONS	27
CHAPTER 3 – SOLUTION’S ARCHITECTURE	29
3.1 – COMMUNICATION STACK CHOICE	33
3.2 – HARDWARE SOLUTION CHOICE.....	34
CHAPTER 4 – SOLUTION’S IMPLEMENTATION	39
4.1 – METHODOLOGY	39
4.2 – IDE CHOICE	39
4.3 – COMMUNICATION MIDDLEWARE DEFINITION AND SETUP	40
4.4 – SETTING UP THE HARDWARE	43
4.5 – SETTING UP THE NANO-GATEWAY	44
4.6 – SIMPLE NODE TO GATEWAY COMMUNICATION	50
4.7 – READING SENSORS ON THE NODE	59
4.8 – STATE MACHINE AND SENDING SENSOR INFORMATION TO THE GATEWAY	62

4.9 – INTRODUCING LOCAL FEEDBACK AT NODE WITH LEDS.....	63
4.10 – CLOUD OR NOT TO CLOUD? AND WHICH ONE TO CHOOSE?.....	65
4.11 – CLOUD INTEGRATION	66
4.12 – DISPLAYING THE PARKING SPACE STATUS AND ALLOWING TO RESERVE REMOTELY USING A SIMPLE INTERFACE	77
4.13 – SENDING THE RESERVED INFORMATION TO THE NODE AND DISPLAYING THE RESERVATION STATUS LOCALLY ON THE NODE.....	82
4.14 – SCANNING USER PROXIMITY AND ALLOWING PARKING ONLY TO THE USER WHO MADE THE RESERVATION	82
4.15 – ROAD BUMPS, PROBLEMS AND QUESTIONS	84
CHAPTER 5 – CONCLUSIONS AND NEXT STEPS.....	87
BIBLIOGRAPHY	89

Tables List

TABLE.2.1: LIST OF PROPOSALS, ITS TITLES, ITS AUTHORS, YEAR OF RELEASE AND MAIN TECHNOLOGIES.....	20
TABLE.2.2: LIST OF EACH PROPOSAL'S ADVANTAGES AND DISADVANTAGES	21
TABLE.2.3: STRONG AND WEAK POINTS FOR EACH PROPOSAL.....	22
TABLE 3.1: COMPARISON OF COMMUNICATION TECHNOLOGIES FOR IoT [22], [25], [26], [27], [28].....	30

Figures List

FIGURE.1.1: EUROPEAN DISABLED PERSON PARKING CARD	2
FIGURE 2.1: PROPOSAL [4] USER INTERFACE IN THE SERVER SOFTWARE.....	8
FIGURE 2.2: PROPOSAL [5] MOBILE PHONE APPLICATION USING HTML5 AND JQUERY, OVERLAYING PARKING INFORMATION WITH GOOGLE MAPS.....	9
FIGURE 2.3: BLOCK DIAGRAM OF GSM MODULE IN PROPOSAL [6]	10
FIGURE 2.4: PROPOSAL [7] GSM MODULE BLOCK DIAGRAM	10
FIGURE 2.5: RESERVATION AND AVAILABILITY IN PROPOSAL [8] MOBILE PHONE APPLICATION	11
FIGURE 2.6: WSN USAGE IN PROPOSAL [9]	12
FIGURE 2.7: PARKING SYSTEM PROPOSED ON [10]	13
FIGURE 2.8: IMAGE PROCESSING AND OBJECT IDENTIFICATION ON PROPOSAL [11]	13
FIGURE 2.9: HARDWARE PROTOTYPE PRESENTED IN PROPOSAL [12]	14
FIGURE 2.10: MONITORING DASHBOARD AVAILABLE IN PROPOSAL [13]	15
FIGURE 2.11: ARDUINO BASED PROTOTYPE IN PROPOSAL [14]	16
FIGURE 2.12: PROPOSAL [15] FRAMEWORK DEFINITION	16
FIGURE 2.13: PROPOSAL [16] ARCHITECTURE	17
FIGURE 2.14: PROPOSAL [17] ARCHITECTURE	18
FIGURE 2.15: SIMULATION SCREENSHOT OF PROPOSAL [18] USING VEHICLE STARTER KIT	19
FIGURE 2.16: SIMON PROJECT CONTEXT	23
FIGURE 2.17: ICT ENHANCED EUROPEAN DISABLED BADGE PROPOSED BY SIMON PROJECT	24
FIGURE 2.18: THE PROPOSED ARCHITECTURE FOR SIMON PROJECT	25
FIGURE 2.19: PHONE PARK RESERVED PARKING SPOTS IN FORUM MONTIJO SHOPPING MALL.....	26
FIGURE 2.20: PARKING SPACE WITH ACCESS BARRIER AND INFORMATION PANEL.	27
FIGURE.3.1: PROPOSED SOLUTION BLOCK DIAGRAM.....	31
FIGURE.3.2: LoRA AND NB-IoT COMPARISON[27]	33
FIGURE.3.3 – NB-IoT ADVANTAGES AND DISADVANTAGES	34
FIGURE.3.4 – LoRA ADVANTAGES AND DISADVANTAGES	34
FIGURE.3.5: PyCOM LoPy 4 SPECIFICATIONS	35
FIGURE.3.6: PyCOM LoPy 4 PINOUT	35
FIGURE.3.7: ADAFRUIT FEATHER 32U4 MAIN SPECS	36

FIGURE.3.8: ADAFRUIT FEATHER 32U4 LoRA PINOUT	37
FIGURE.4.1 – LORIIOT INFRASTRUCTURE FOR IoT	41
FIGURE.4.2 – LORIIOT NETWORK SERVER FEATURES	42
FIGURE.4.3 – THE LoRAWAN PROJECT APPROACH PROPOSED BY TTN	43
FIGURE.4.4 – PYCOM EXP BOARD 3.0 HARDWARE	43
FIGURE.4.5 – LoPy4 WITH THE ANTENNA CONNECTED	44
FIGURE.4.6 – THE TTN CONSOLE.....	44
FIGURE.4.7 – GATEWAY REGISTRATION INPUT FORM	45
FIGURE.4.8 – GATEWAY ID GENERATOR	46
FIGURE.4.9 – GATEWAY DETAILS AFTER REGISTRATION – FIRST HALF OF SCREEN	47
FIGURE.4.10 – GATEWAY DETAILS AFTER REGISTRATION – SECOND HALF OF SCREEN	48
FIGURE.4.11 – GATEWAY UP AND RUNNING ON A BREADBOARD	49
FIGURE.4.12 – GATEWAY CONNECTED AND EXCHANGING MESSAGES WITH TTN.....	50
FIGURE.4.13 – ADDING A NEW APPLICATION ON TTN CONSOLE.....	50
FIGURE.4.14 – APPLICATION OVERVIEW AND EUIS.....	51
FIGURE.4.15 – REGISTERING A NEW DEVICE FOR THIS APPLICATION	51
FIGURE.4.16 – INITIALIZATION BLOCK.....	52
FIGURE.4.17 – JOIN BLOCK	53
FIGURE.4.18 – TEST BLOCK.....	53
FIGURE.4.19 – NODE IN DEBUG CONNECTED THROUGH MICRO USB PORT	54
FIGURE.4.20 – UPLOAD OF NODE SOFTWARE AND DEBUG MESSAGES DURING THE JOIN PHASE.....	55
FIGURE.4.21 – ACTIVATION PACKAGE ON THE COMMUNICATION INITIALIZATION ON TTN CONSOLE	56
FIGURE.4.22 – COMMUNICATION FLOW (UPLOAD) AND THE RESPECTIVE PAYLOAD.....	57
FIGURE.4.23 – ON THE TTN CONSOLE SENDING DATA TO THE NODE OR TO THE GATEWAY CAN BE SIMULATED	58
FIGURE.4.24 – CHECK THE RECEIVE ON THE GATEWAY	58
FIGURE.4.25 – CHECK THE RECEIVE ON THE NODE (THROUGH DEBUGGING CONSOLE ON ATOM)	59
FIGURE.4.26 – HONEYWELL HMC5883L PINOUT (I ² C)	59
FIGURE.4.27 – SENSOR CONNECTIONS TO THE LoPy 4	60
FIGURE.4.28 – POSITIONING READINGS FROM THE SENSOR – THREE AXIS, HEADING AND DECLINATION	61
FIGURE.4.29 – NODE WITH SENSOR.....	61
FIGURE.4.30 – STATE MACHINE CODE	62
FIGURE.4.31 – MESSAGE SENT TO THE GATEWAY – NOT OCCUPIED STATE	63
FIGURE.4.32 – THE PARKING SPACE IS NOW OCCUPIED	64
FIGURE.4.33 – STACK NAME INPUT.....	67
FIGURE.4.34 – STACK PARAMETERS	68
FIGURE.4.35 – STACK PARAMETERS CONTINUED	69
FIGURE.4.36 – STACK CAPABILITIES ACKNOWLEDGE.....	70
FIGURE.4.37 – STACK CREATION IN PROGRESS	70
FIGURE.4.38 – STACK CREATION COMPLETE	70
FIGURE.4.39 – MANAGE THINGS ON AWS IoT.....	71
FIGURE.4.40 – MQTT CLIENT FOR TESTING ON AWS IoT.....	71
FIGURE.4.41 – EXAMPLE OF A MESSAGE SENT FROM THE NODE (EMPTY PARKING SPACE)	72
FIGURE.4.42 – DOWNLINK MESSAGE FORMAT	72
FIGURE.4.43 – SEND MESSAGES WITH DYNAMODB AS TARGET	73

FIGURE.4.44 – CREATING A DYNAMODB TABLE FOR STORING PARKING SPACES MESSAGES	73
FIGURE.4.45 – ADDING A ROLE FOR AWS IoT ACTION TO WRITE IN DYNAMODB	74
FIGURE.4.46 – FINISHING THE RULE CREATION	75
FIGURE.4.47 – UPLINK MESSAGES FROM THE NODE INSERTED AT DYNAMODB.....	76
FIGURE.4.48 – LAMBDA FUNCTION CREATION	76
FIGURE.4.49 – LAMBDA FUNCTION CREATED AND READY TO BE CONFIGURED	77
FIGURE.4.50 – AWS LAMBDA CODE WINDOWS WITH NODE.JS CODE FOR SCANNING THE DYNAMODB TABLE	77
FIGURE.4.51 – AWS LAMBDA TEST EXECUTION RESULTS	78
FIGURE.4.52 – CREATING A NEW API IN AWS API GATEWAY.....	78
FIGURE.4.53 – DEFINING GET METHOD AND ASSOCIATING WITH THE LAMBDA FUNCTION	79
FIGURE.4.54 – DEPLOYING THE API IN BETA STAGE	79
FIGURE.4.55 – REST API WITH PARKING SPACE INFORMATION.....	80
FIGURE.4.56 – S3 BUCKET PROPERTIES	80
FIGURE.4.57 – ENABLING STATIC WEBSITE HOSTING	81
FIGURE.4.58 – WEB INTERFACE FOR RESERVED PARKING VALIDATION PROJECT STATUS	81
FIGURE.4.59 – NODE LISTENING TO MESSAGES FROM THE GATEWAY	82
FIGURE.4.60 – CHECKING THE MESSAGE PAYLOAD AND SENDING ACKNOWLEDGE	82
FIGURE.4.61 – SCANNING BLUETOOTH DEVICES AND SHOWING MAC ADDRESS FOR EACH ADVERTISEMENT	83
FIGURE.4.62 – CHECKING CORRECT USER PRESENCE AND “OPENING” THE BARRIER	83
FIGURE.4.63 – SETTING PIN 11 TO GPIO MODE AND AS AN OUTPUT	84
FIGURE.4.64 – ERROR ON ATOM USING PYMAKR PLUGIN.....	84

Acronyms List

ABP	Authentication By Personalization
AI	Artificial Intelligence
API	Application Programming Interface
BLE	Bluetooth Low Energy
BSS	Business Support System
CLI	Command Line Interface
DR	Demand Response
EMEL	Empresa Municipal de Estacionamento de Lisboa
GPIO	General Purpose Input Output
GSM	Global System for Mobile (Communications)
GUI	Graphical User Interface
HLD	High-Level Design
HSV	Hue, Saturation, Value Color Model for Images
HTML5	HyperText Markup Language version 5
HTTP	HyperText Transfer Protocol
I2C	Inter Integrated Circuit
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JSON	JavaScript Object Notation
LED	Light Emitting Diode
LoRa	Long Range

LPWAN	Low Power Wide Area Networks
M2M	Machine to Machine
NB-IoT	Narrow Band Internet of Things
NFC	Near Field Communication
OSS	Operational Support System
OTAA	Over The Air Authentication
PaaS	Platform as a Service
PoC	Proof of Concept
QoS	Quality of Service
QR Code	Quick Response Code
REST	REpresentation State Transfer
RFID	Radio-Frequency Identification
RGB	Red, Green, Blue Color Model for Images
SFM	Sensor and Feedback Module
SMS	Short Message Service
TTN	The Things Network
USB	Universal Serial Bus
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

1

Chapter 1 – Introduction

This chapter introduces the problem and tries to give some clues why this problem keeps affecting our society. It also talks about the motivation for this dissertation and why a technological solution is a path to solve this issue. Finally, it lists the structure of the document.

1.1 - Context

In 2007 one study[1], revealed that: “45% of total traffic is cruising for a parking space; 64% of local traffic is cruising for a parking space and nearly 1 in 6 parked vehicles parks illegally, with illegal parking increasing exponentially as the curb fills up”.

Every day, a city like Lisbon, receives in excess of 370 000 vehicles from outside the city, to join the already huge number of cars from the city’s habitants. And considering the number of available non-paid parking spaces, it’s easy to understand the motivation why, some drivers, choose to not respect the law and park in disabled person reserved parking spaces.

Some misinformation, also explains why, some drivers, think that using this parking places, with an excuse like “it’s just 5 minutes” does not constitute an infraction, and that they are not causing inconvenience to anyone.

Also, the proliferation of fake badges is something that some users are resorting, to be able to park for free, and also get access to this reserved parking spots, that are usually better located, to make an easier access for a disabled person, to public and private services. There are also people selling legit badges, stolen or that they don’t use anymore, in the classified websites, trying to profit from this situation.

In the last years, law and public services tried to reduce the incidence of this problem: by either investing in awareness campaigns, more priority of the police forces focus in deterring and fining drivers who do not respect the law, and also by raising the value of fines and an additional penalty by subtracting 2 points (from a starting number of 12

points) from the driver's license, for anyone who parks in these places, without the legit blue badge.



Figure.1.1: European Disabled Person Parking Card

1.2 – Motivation and Objectives

The main goal of this dissertation, is not only to propose a viable, cost effective, and usable solution for this problem, but also raise awareness for this problem. Handicap people did not choose to be in this situation. By reserving them parking spaces, near public and private services and utilities, they are given more freedom, so they can be and feel useful in our society. It is a way to give them more resources, so they can live their lives, not submitting them to home reclusion.

All the conditions discussed in the 1.1 section, are not the only reasons why this problem subsists. Not only the occupancy of the reserved parking constitutes a problem. Lisbon, and other big cities as well, are facing other huge traffic problems, caused by phenomena like double parking, crosswalk parking and sidewalk parking. People are always very absorbed in their own lives. The everyday pressure and busy schedules are also big factors leading to these behaviors.

1.3 - Document Structure

Following this first chapter dedicated to introducing the problem and looking at the motivation and objectives, the second chapter is dedicated to the state of the art, showing proposed, and already implemented solutions, and understanding what's being done to solve this problem.

On the third chapter the main focus is on the architecture of the proposed solution, showing the several options, regarding technologies, and the fourth chapter is about the implementation of this solution, and why certain decisions were made.

Finally, the fifth chapter covers the conclusions and suggests some possible next steps for this solution.

2

Chapter 2 – State of the Art

By taking a look at the last decade's posts and papers in the research community, it is easy to see that there are already a lot of solutions being proposed, mainly in the academic field, with some of them candidates to implementation, either via the launch of a commercial product, or through some financing program or being chosen to real-world trials with real users in live environments.

In this chapter some of these proposals are presented, especially the ones that are more robust, or that have a bigger appeal to become a real-world solution, or that are already in trial status. Also are presented some commercial solutions, that have already been implemented.

The main focus is to illustrate what each solution brings to the table as an innovation, and how it achieves the purpose of keeping the parking spaces only occupied by people who have the validation.

2.1 – Smart Cities and the Internet of Things

Lots of these research works are leaning into a relatively new concept field, which are the smart-cities, because of the big affinity between the large gathering of data from these cities, via arrays of sensors and other devices, all networked, providing information about its users and vehicles, and the implementation of technologies that benefit those same users, reducing traffic, delays and pollution.

One definition of smart-cities is “connecting the physical infrastructure, the IT infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city.” [2]

A big field of study that these smart cities are referencing are the gathering of data from parking lots, available parking spaces, and traffic, to better reference the users to a certain destination, that allows them reach destination, using the fastest path, and to park quickly and efficiently, without causing delays in traffic or causing the users to keep roaming

around a given area until they find a parking space, increasing traffic, fuel consumption, and rising the levels of pollution.

Another area of study that these proposals are focusing, is the usage of Internet of Things (IoT) solutions. IoT is a paradigm that proposes the application of heterogeneous devices such as home-appliances, consumer electronics, sensors, vehicles and others and defines the way they communicate and the interconnections between them, with the goal to exchange and aggregate data from them, in benefit of their end users.

According to [3], which explains the usage of the IoT paradigm in the smart cities context, there are several challenges that need to be addressed, before this solutions move from prototypes and trials, to become definitive solutions, able to help solve the problems of our modern cities:

- Security and privacy: all the data being exchanged between devices need to be protected (encrypted) so it cannot be intercepted by hackers. Also, all this data might constitute a privacy problem, if someone is able to cross information or identify user specific data;
- Heterogeneity: given all the multiplicity of technologies, protocols and standards, its and herculean effort to put all this multitude of devices communicating and working together;
- Reliability: some of these devices are already of industrial standard and reliability and conform with most of the requirements of public usage, but still need to account with vandalism, harsh weather conditions, temperature and humidity oscillations, etc.;
- Large scale: when these architectures are scaled to accommodate the coverage of a metropolitan city, the number of devices to install and maintain, and all the energy and network connections will become huge requirements in the planning, implementation and maintenance phases, to make the system function at its full performance;
- Legal and social aspects: this is another huge barrier, because of the volume and specificity of the data, the collector (and keeper / owner) needs to conform with all internal and foreigner laws and to make sure that the users consent in the usage of their data;
- Big data: not only the collection and repository of these huge amounts of data is a challenge, but also the algorithms and computational power to crunch it, and return usable data to take actions, and increase services and methods performance;

- Sensor networks: another big aspect of IoT environments, this one goes hand in hand with reliability, and the capacity of the sensors to perform at their best, with residual down-times and huge accuracy rates – also the capacity of working as nodes, broadcasting and relaying information to the other nodes, is a huge factor weighting in the choice of these networks;
- DR barriers: which the author divides in three major barriers: consumers', producers', and structural barriers.

2.2 – Proposals presentation

Taking all the knowledge from the previous section, the objective for this chapter, become the understanding of the several proposed solutions, and try to bring at least one example for each technology, to be able to compare the advantages and disadvantages in choosing it.

The proposals are listed in chronological order, from the oldest to the newest one, considering the release year.

The proposal “*Design and implementation of a street parking system using wireless sensor networks*” [4] is based on the implementation of WSNs, in which the sensor used to detect the occupancy of a parking space by a car, is of the magnetic type. It uses a state machine to conclude about the occupancy based on the sensor signals. The communication stack implemented is ZigBee which also accounts for a good energy consumption.

This proposal is already in trial, in a controlled environment, with 62 parking spaces using the system devised in the proposal. Although very occupancy oriented, it was chosen because of the magnetic sensor choice – which is a very cheap sensor – and how the authors accounted for the sensor read values deviation, influenced by ambient temperature fluctuation, and how this was corrected by the proposed algorithm. This system is also very energy efficient and the final results achieved through numerous testing, in different conditions, and layouts, show an accuracy superior to 99%.

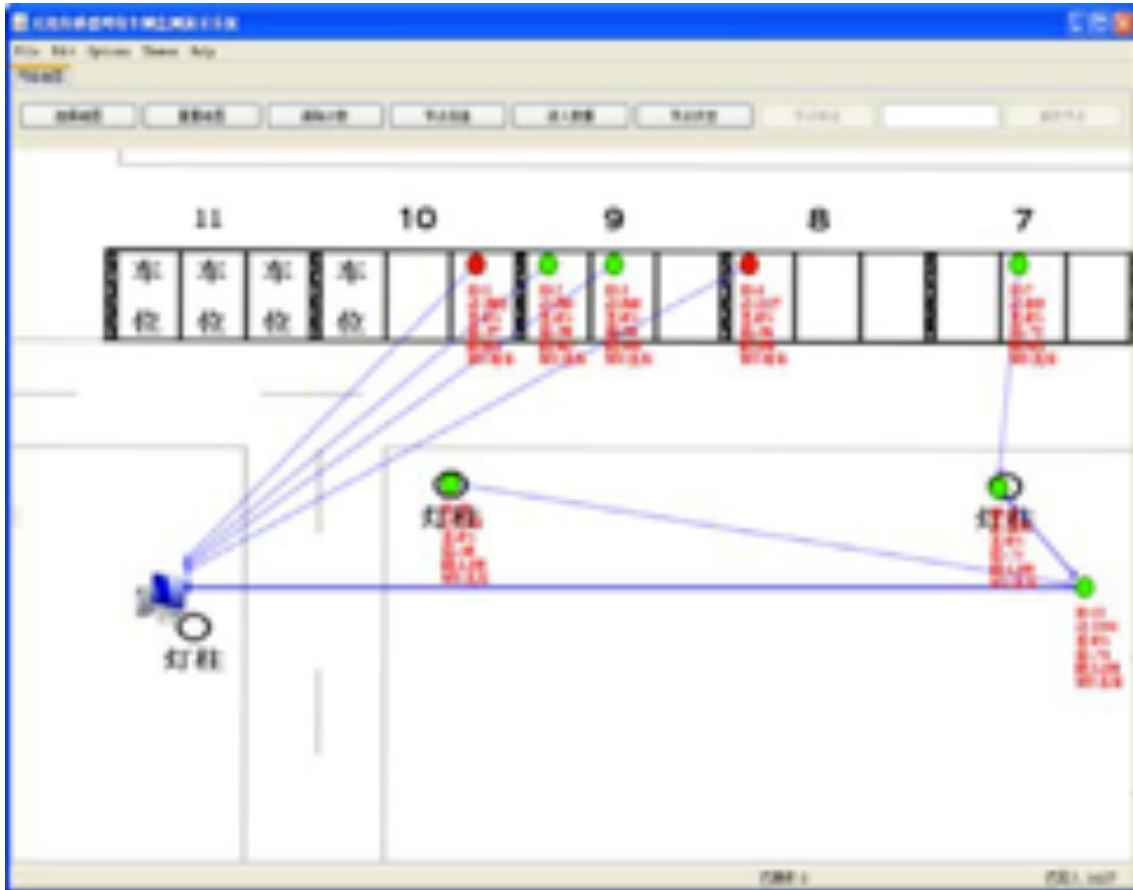


Figure 2.1: Proposal [4] user interface in the Server software.

The next proposal, “*Smart parking service based on Wireless Sensor Networks*” [5], also makes usage of the WSN concept, with the communication stack chosen also ZigBee, but the occupancy detection, is in this case based in a light sensor.

The extended usage of WSN, allow to achieve a very modular network, capable of scaling as demands arise. It’s also very occupancy oriented but proposes a mechanism of remote feedback to the end users, through the implementation of mobile applications. It is based in a very modular approach with 4 layers: WSN, Embed Web Server, Central Web Server and Mobile Application. The usage of very modern technologies (for the time it was proposed) like HTML5 and jQuery, overlaying the information on Google Maps, allows for adaptive web pages that can be visualized correctly in every mobile device, without usability constraints.

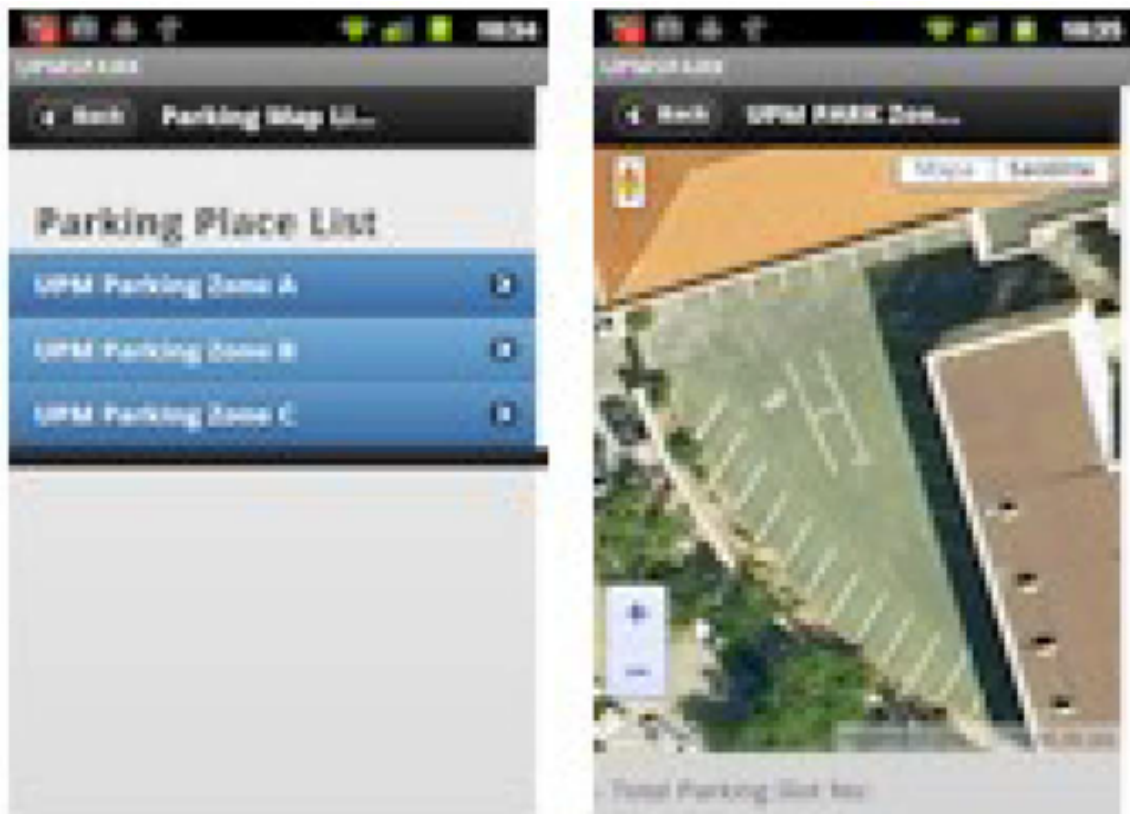


Figure 2.2: Proposal [5] mobile phone application using HTML5 and JQuery, overlaying parking information with Google Maps.

“A Secure Parking Reservation System Using GSM Technology” [6] is the first proposal on the list to propose the concept of reservation, and also the first in the list to make use of GSM technology. Very simple although very well structured, is based in two distinct modules: Security Reservation Module, that through the usage of password imposes a high level of security, and Parking Lot Monitoring Module, that implements the logic between reservation, access and allocation of parking spaces.

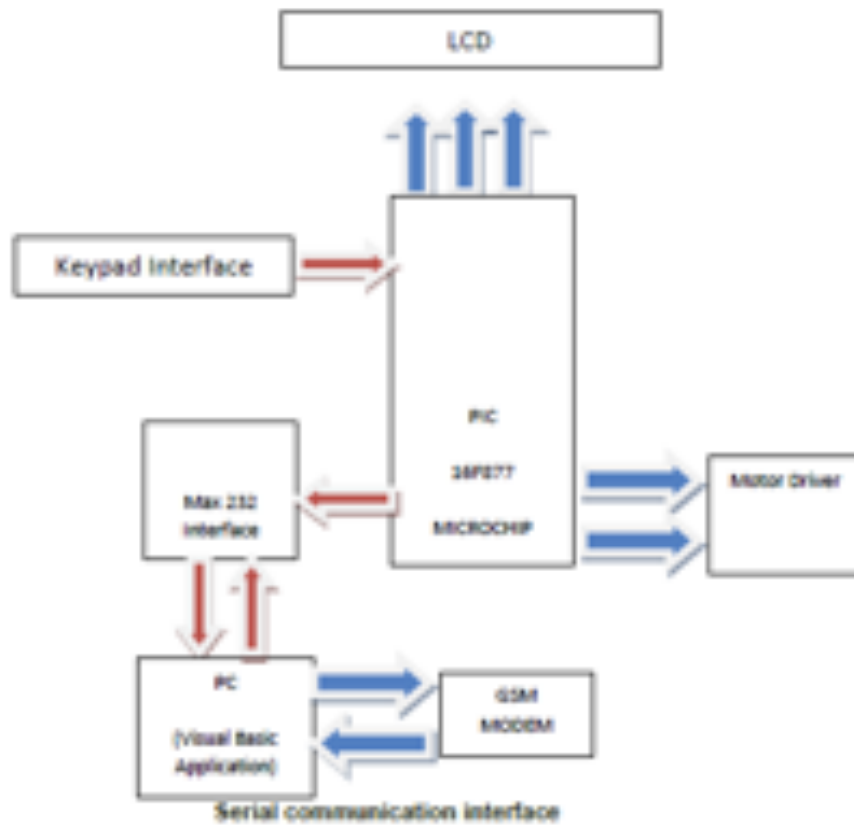


Figure 2.3: Block diagram of GSM module in proposal [6]

“An Intelligent Parking Guidance and Information System by using image processing technique” [7] is the first on the list to use image acquisition and processing to identify parking spaces allocation with vehicles. It is very focused on the image processing and vehicle identification part. It not very modular, and it does not give much clues about how to scale the solution. It’s also GSM based in the usage that makes of SMS to allow reservation and give feedback to end users.

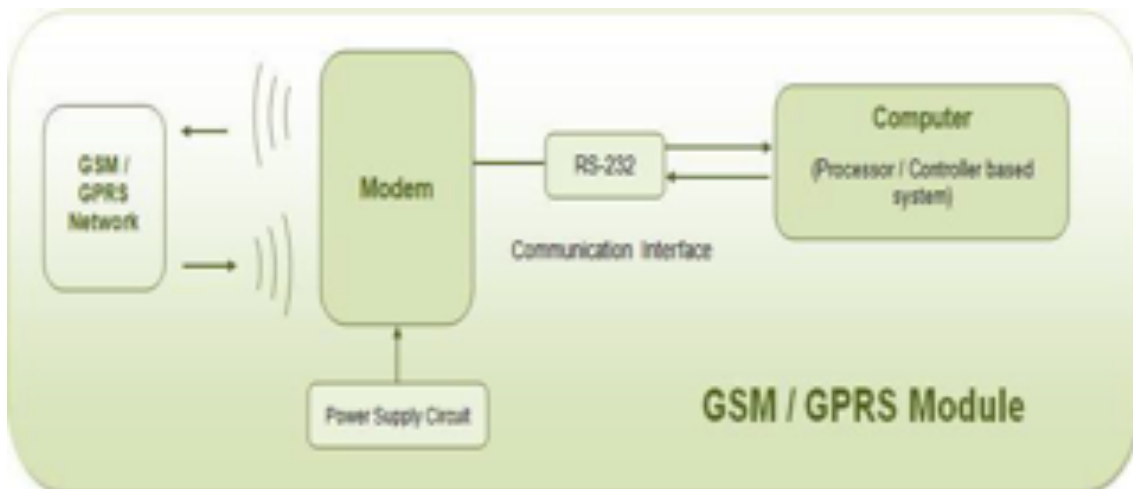


Figure 2.4: Proposal [7] GSM module block diagram

“DisAssist: An internet of things and mobile communications platform for disabled parking space management” [8] is the only proposal in this list, that is completely focused on the disabled parking slots validation, reservation and allocation. It gathers knowledge from IoT and Smart Cities paradigms, using it to automate the detection of reserved parking spaces, through the usage of M2M, with the communication based in the ZigBee standard (IEEE 802.15.4).

Another big achievement of this proposal is the capacity for a user to reserve a space in advance, and afterwards it guides the user to the reserved spot, without the risk of someone overtaking the parking space, while in route. By providing a mobile app, this proposal, also raises the bar regarding usability, giving the users a guided navigation to the chosen parking space, while also giving local feedback through very simple colored lights. Because the authorization depends of a unique ID it’s also very secure and reliable. It’s also one of the only proposals where the user anonymity is debated and pursued.



Figure 2.5: Reservation and availability in proposal [8] mobile phone application

The proposal *“Wireless Sensor Network and RFID for Smart Parking System”* [9] is another proposal based on WSN, offering a very modular approach on the architecture. Although it does not present an actual implementation, it is in its concept very cost effective,

through a very advanced and well thought usage of WSN nodes and the way they communicate and relay information to other nodes, with the goal to reach the gateway. The nodes are divided in three types: monitoring nodes to which the sensors are connected; routing nodes, that forward the information, hop to hop, from the monitoring nodes, through the usage of a tree topology; and finally, the sink node, that interfaces this information to the management center. It's another proposal much more occupancy and access oriented. It's the first on the list to make usage of RFID technologies, to univocally identify a car when checking in and checking out, allowing a fast and safe payment process.

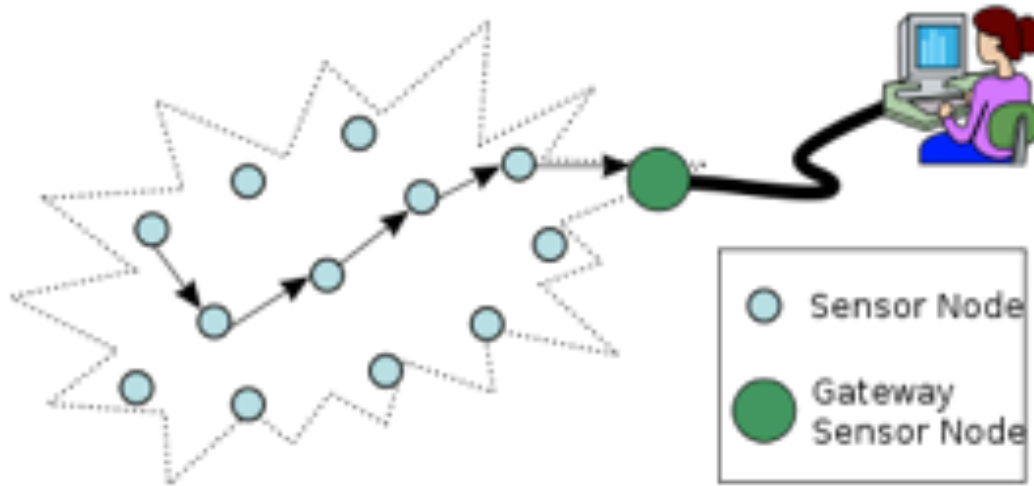


Figure 2.6: WSN usage in proposal [9]

“Automated Parking System With Bluetooth Access” [10] presents a solution based on an automated silo, for parking purposes. Although this is not the focus of this dissertation, this proposal is presented because of the simplicity and cost effectiveness of the validation/occupancy solution based on the Bluetooth communication stack. On the other side the costs of implementation of an automated parking silo are still huge by today's standards, but already implemented in some cities, mainly in the US.

An automated silo parking is excellent for the end user, because it's very secure, and as the parking is done automatically – the user just needs to stop the car in a certain spot and leave it – there no risks to damaging its own car and the other cars. As weaknesses of this system, if several users arrive at the same time, it is not very efficient, as the users need to wait for each previous driver's car to be parked before it allows them to park.



Figure 2.7: Parking system proposed on [10]

Next, “*Intelligent Parking Management System Based on Image Processing*” [11] only focus on the parking space availability and is also heavily associated with closed parking solutions. It devotes most part of the paper to the image capture and processing, and vehicle identification phase. The resulting algorithm is excellent and allow the usage of a very simple and cost-effective wireless camera.

How does the system work? First, an image of the empty park, is taken to account as reference of the location of the empty parking spots, designated by green circles drawn at each spot. Then the image is converted from the RGB model to the HSV model. Then the HSV image is converted to a grey scale format, and afterwards to black and white taking in account where the threshold is above 70%. Next, by using erosion and dilation in the image, small parts that are white, are automatically removed.

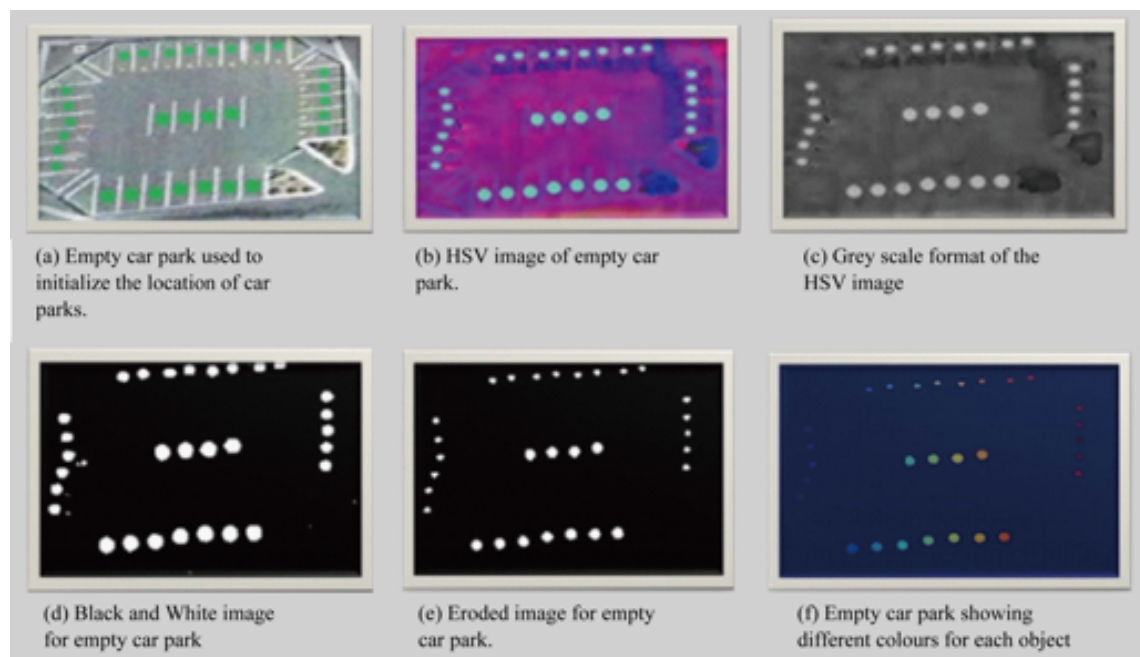


Figure 2.8: Image processing and object identification on proposal [11]

“Monitoring Parking Space Availability via Zigbee Technology” [12] is a proposal in which the identification of occupancy is based on an infrared sensor, which is very cheap, but also error prone, with huge instability caused by certain lightning sources, although the author refers that the proposed distance (30cm) of the sensor minimizes these effects. Also raises some barriers in its usage in open street parking. It pursues and achieves a very good energy efficiency. Also proposes a modular architecture with the connection between modules achieved through the usage of ZigBee. The GUI display is very PoC oriented, needing a revamp for end users’ acceptance.

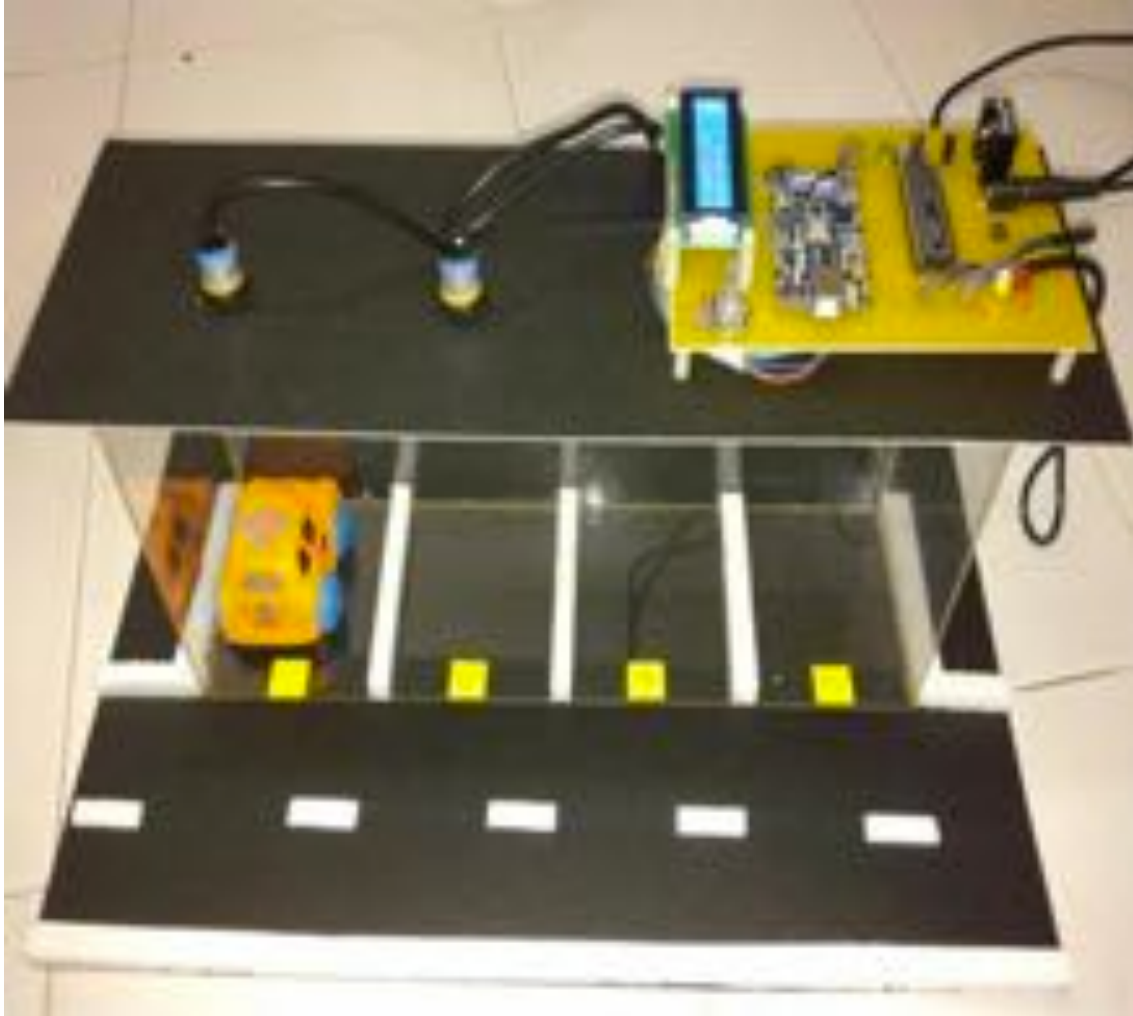


Figure 2.9: Hardware prototype presented in proposal [12]

“Rapid development of smart parking system with cloud-based platforms” [13] shows a much more modern approach to the architecture. Is the first in the list to use the Cloud Services in a PaaS configuration, with a very good modular and well thought out architecture, with good interoperability (openness) as it makes use of JSON and REST services to expose its data. It is also, more occupancy inclined, as the majority of the presented proposals.

Other of the innovations that this proposal adheres to, is the usage of a modern dashboard, interfacing data to graphics and visual representation, to allow not only to understand the

occupancy of each parking lot, but also to monitor other parameters of the system, in a very infographic perspective, and to have an overall perspective of the system in a fast glance. One of the available parameters is the battery state of each WSN, as all the sensor nodes are battery powered, with a time frame of 2 years for battery duration. The communication stack is once again ZigBee, with all the communication and power consumption performance associated.



Figure 2.10: Monitoring dashboard available in proposal [13]

“A wireless smart parking system” [14] presents another modular architecture that makes usage of RFID tags to control the payment. It allows guidance to the parking spot within a 2km radius, through the usage of a mobile device application. There is not much information about the mobile application that supports the service and there is no definition about the sensor used after the PoC – during the PoC the presence of cars was simulated by sending a signal after a certain amount of time.

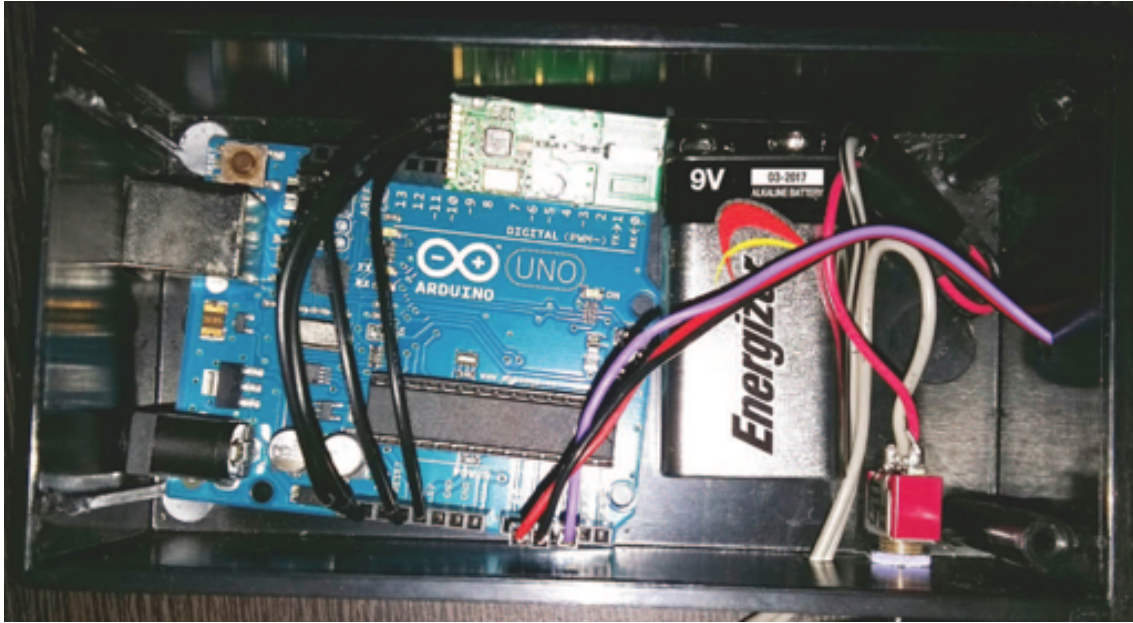


Figure 2.11: Arduino based prototype in proposal [14]

“Car park management with networked wireless sensors and active RFID” [15] presents an excellent framework definition and a very good modularity. This modularity allows to multiple configurations to be achieved in the implementation, using always the same system, although it leans more on the occupancy trend. It’s also very cost aware, as it uses an ultrasonic sensor for occupancy capture. It combines the usage of WSNs and RFID technology. In certain scenarios the RFID tag being active, can be used to identify the car, and guide it to its reserved parking lot. It can also accomplish the service “Where did I park my car” if the user forgets the spot, where he parked the car and, in certain situations, to check for improper parking of an unauthorized car in a reserve spot, or also as a contactless paying system, if needed.

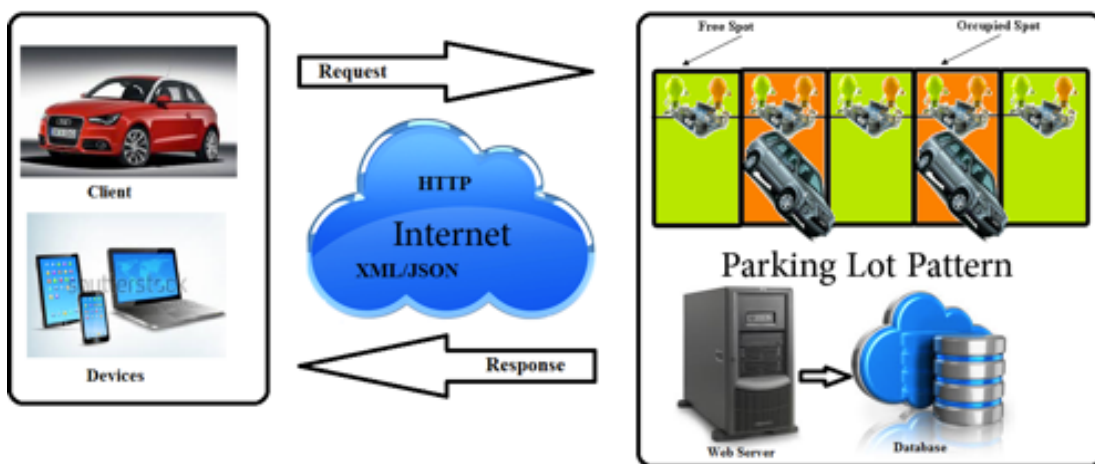


Figure 2.12: Proposal [15] Framework definition

“Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies“ [16] is a “think out of the box” approach to the

parking problems. It's also one of the few to focus on street parking instead of closed parking lots. Its approach is based on a mobile device usage, and all the data that the device's sensors can report, using Bluetooth and Wi-Fi Direct to communicate. This way it devises an algorithm capable of understanding when the user is walking or driving, and also when is leaving the parking space, showing it available for other users.

This way it renders needless the installation of external sensors making this a very cost-effective proposal. The only major doubt is about the mobile device battery consumption, with all the sensor usage and communication (information not given/investigated in the proposal).

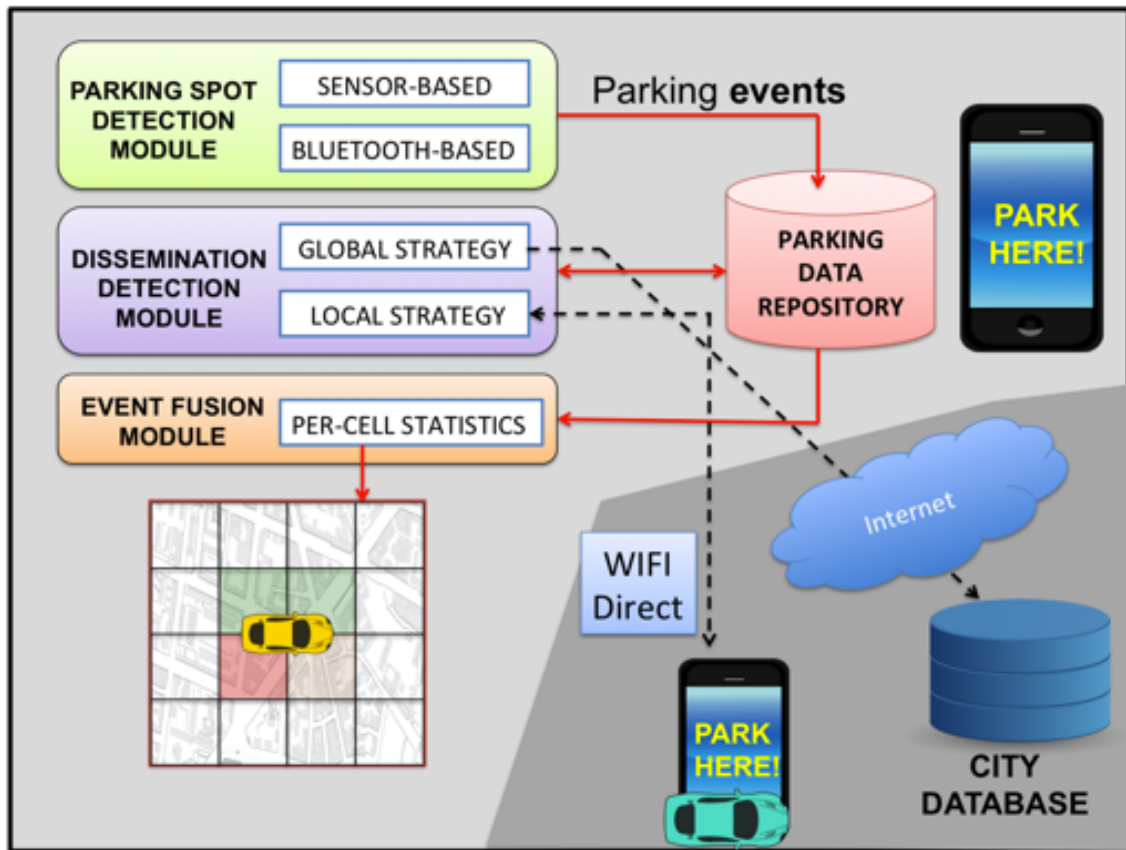


Figure 2.13: Proposal [16] architecture

“A Prototype for IoT based Car Parking Management system for Smart cities” [17] uses RFID embed in the vehicle license plate to achieve the identification phase. This way accomplishes a very secure identification, also allowing to report a stolen car that arrives for identification. This is a system that raises a lot of constraints on the privacy side, and also needs at least nationwide implementation. Its more positioned for closed parking, and relies on infrared sensors for occupancy, which can be error prone. The proposed system is very modular and allow reservation through an online booking module.

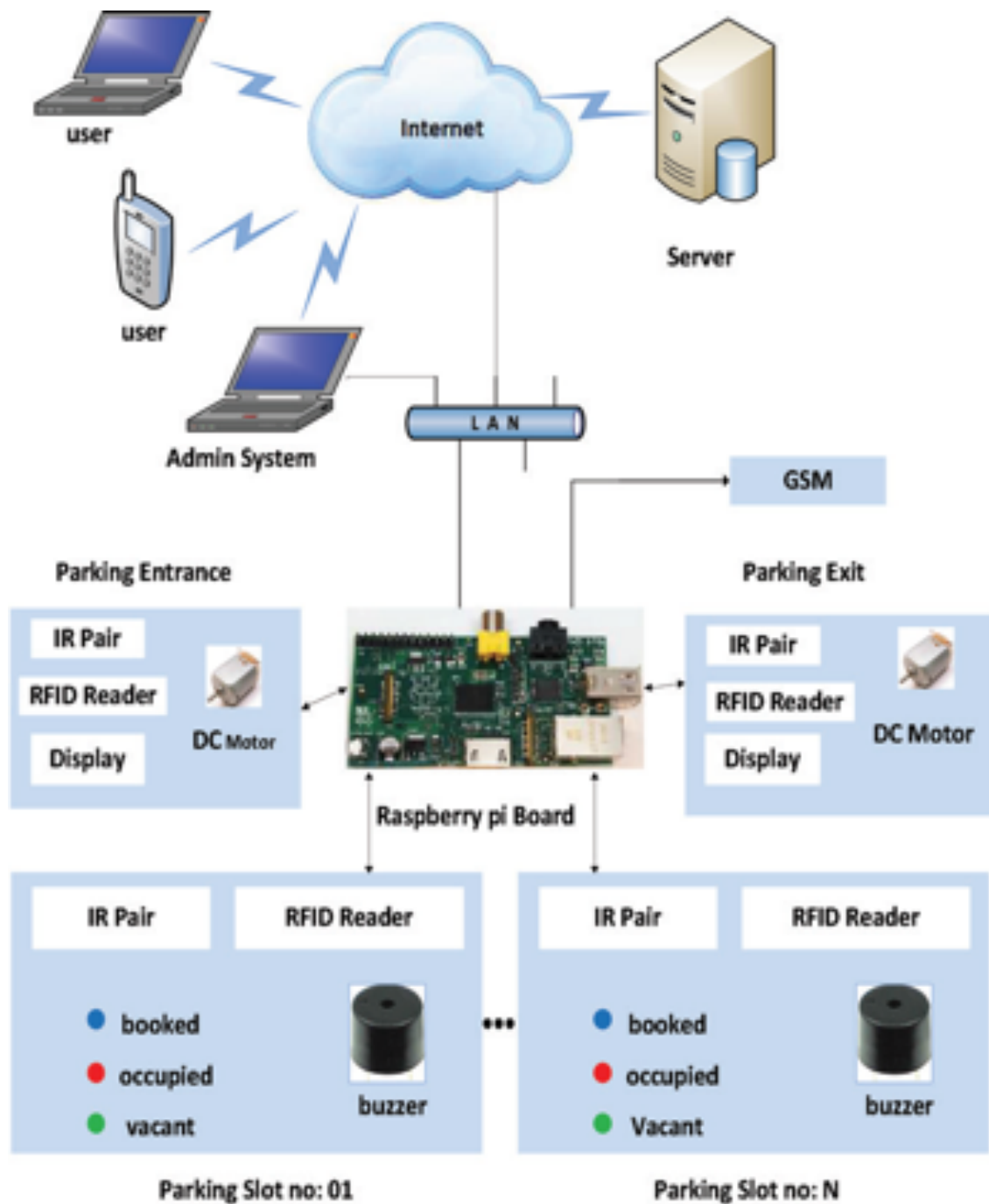


Figure 2.14: Proposal [17] architecture

“BlueParking: An IoT based Parking Reservation Service for Smart Cities” [18] is very modular in its inception. It’s also very well defined and makes use of some new concepts of Cloud computing as PaaS, Traffic Estimator and Path Finder Service. It also adopts a very recent (and not yet implemented in a real environment as of today or even standardized) concept which is the Connected Car. Although there are already some developments and trials in this path, the proposal is very dependent of this development. It allows reservation, guidance to the parking spot and uses traffic estimation to choose the best itinerary.



Figure 2.15: Simulation screenshot of proposal [18] using vehicle starter kit

Reference number	Paper title	Authors	Year	Main Technologies
[4]	Design and implementation of a street parking system using wireless sensor networks	J. Gu, Z. Zhang, F. Yu, and Q. Liu	2012	Magnetic Sensor ZigBee
[5]	Smart parking service based on Wireless Sensor Networks	J. Yang, J. Portilla, and T. Riesgo	2012	Light Sensor ZigBee
[6]	A Secure Parking Reservation System Using GSM Technology	Y. Rahayu and F. N. Mustapa,	2013	GSM/SMS
[7]	An Intelligent Parking Guidance and Information System by using image processing technique	P. DharmaReddy, A. RajeshwarRao, and D. S. M. Ahmed	2013	Image detection and processing GSM/SMS
[8]	DisAssist: An internet of things and mobile communications platform for disabled parking space management	L. Lambrinos and A. Dosis	2013	Magnetic Sensor ZigBee GSM
[9]	Wireless Sensor Network and RFID for Smart Parking System	M. Patil and V. N. Bhonge	2013	RFID ZigBee
[10]	Automated Parking System With Bluetooth Access	H. Singh, C. Anand, V. Kumar, and A. Sharma	2014	Bluetooth
[11]	Intelligent Parking Management System Based on Image Processing	H. Al-Kharusi and I. Al-Bahadly	2014	Image detection and processing WiFi
[12]	Monitoring Parking Space Availability via Zigbee Technology	H. Chien Yee and Y. Rahayu	2014	ZigBee Infrared Sensor
[13]	Rapid development of smart parking system with cloud-based platforms	Z. Suryady, G. R. Sinniah, S. Haseeb, M. T. Siddique, and M. F. M. Ezani	2014	Magnetic sensor ZigBee WiFi
[14]	A wireless smart parking system	O. Orrie, B. Silva, and G. P. Hancke	2015	RFID WiFi
[15]	Car park management with networked wireless sensors and active RFID	E. Karbab, D. Djenouri, S. Boulkaboul, and A. Bagula	2015	Ultrasonic sensor RFID WiFi
[16]	Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies	R. Salpietro, L. Bedogni, M. D. Felice, and L. Bononi	2015	WiFi-Direct Bluetooth Mobile phone sensors
[17]	A Prototype for IoT based Car Parking Management system for Smart cities	B. M. Kumar Gandhi	2016	Infrared sensor RFID
[18]	BlueParking: An IoT based Parking Reservation Service for Smart Cities	M. Amin, R. Kawaguchi, N. Shirmohammad, and M. Sato	2016	IoT Connected Car

Table.2.1: List of proposals, its titles, its authors, year of release and main technologies

2.3 – Advantages and disadvantages resume

The method chosen for analyzing the several proposals, was to resume the major advantages and disadvantages, from a macro point of view.

Reference number	Advantages	Disadvantages
[4]	Excelent energy efficiency Good accuracy Very solid algorithm Temperature flutuations correction	Proposal focus on closed parking More occuppency oriented than validation oriented
[5]	Modular architecture Availability feedback to the final users	Proposal focus on closed parking More occuppency oriented than validation oriented
[6]	Allows reservation	Proposal focus on closed parking More occuppency and security oriented than validation oriented
[7]	Allows reservation	Not very modular More occuppency oriented than validation oriented
[8]	Very modular Handicap oriented Navigates users to available slots Good local and remote feedback Deals with end users' privacy	Needs previous registration Reservation for future time slots not yet developed Because there are no future reservation users might be directed to a slot no longer available
[9]	Very modular Perfect use of tree sensor node network architecture Cost oriented solution	Proposal focus on closed parking Does not present a implementation
[10]	Very cost effective (validation/occuppency) Very simple to park Good energy efficiency	Proposal focus on automated silo parking Very high implementation costs (silo parking) Cannot park several cars at the same time (silo parking)
[11]	Excelent image processing study and algorithm	Proposal focus on availability only
[12]	Modular architecture Good energy efficiency Low implementation cost	Detection is error prone More occuppency oriented than validation oriented
[13]	Cloud based Excelent architecture definition Modular implementation Interoperable as it uses REST and Json to interconnect to other systems	More occuppency oriented than validation oriented
[14]	Modular architecture Allows guidance to an available spot (within 2 km)	More occuppency oriented than validation oriented No detection sensor defined after POC Not much info about mobile application
[15]	Modular architecture Good framework definition Can be deployed in multiple configurations Cost awareness	More occuppency oriented than validation oriented
[16]	Street parking focused No need to install spot sensors Very different approach	It might consume mobile battery very fast (not tested)
[17]	Allows reservation Very Modular	More occuppency oriented than validation oriented Proposal focus on closed parking Detection is error prone
[18]	Modular architecture Cloud based Traffic estimation Allows guidance to available spot Allows reservation	No real implementation on the connected car side (only via simulator)

Table.2.2: List of each proposal's advantages and disadvantages

Next, each proposal was segmented and analyzed, according to its solution and contribution in each of the following areas:

- Detection
- Guidance
- Validation
- Architecture
- Modular design and scalability
- Feedback
- Reservation
- Implementation
- Applicability
- Usability
- Security and privacy
- Economics.

For each of these areas, there is a classification: a plus sign if the proposal addresses correctly that area and a minus sign if it does address that area in a deficient way. No classification means that the proposal is neither strong nor weak in that area.

Reference number	Detection	Guidance	Validation	Architecture	Modular design and scalability	Feedback	Reservation	Implementation	Applicability	Usability	Security and privacy	Economics
[4]	+		-						-			+
[5]			-		+	+			-			
[6]			-				+		-			
[7]			-		+		+		-			
[8]		+	+		+	+	-		+	+/-	+	
[9]				+	+			-	-			+
[10]									-	+/-	+	+/-
[11]	+		-						-			
[12]	-		-		+				-			+
[13]				+	+				-			
[14]	-	+			+				-	-		
[15]				+	+				+/-			+
[16]	+		+	+	+				+			+
[17]	-				+		+		-			
[18]		+		+	+		+	-				

Table.2.3: Strong and weak points for each proposal

Beyond these 15 proposals that more or less cover the spectrum of proposals available, there are two other solutions being presented. One is a commercial service that has several

implementations worldwide, and that served as a motivation for the author picking this dissertation work, and SIMON which is a project sponsored by the European Commission, and currently is in its trial phase, with handicap end users, in the cities of Lisbon, Madrid, Parma and Reading.

The main reason SIMON was not included in the proposals list, is because this project goes far beyond the simple validation of reserved parking, dealing with other subjects, afar the purpose of this dissertation.

SIMON, as described by its proponents in [19], “is a demonstration oriented project with three large scale pilots in Madrid, Lisbon and Parma, aiming to use ICT services to promote the independent living and societal participation of mobility impaired people in the context of public parking areas and multiple transport modes”.

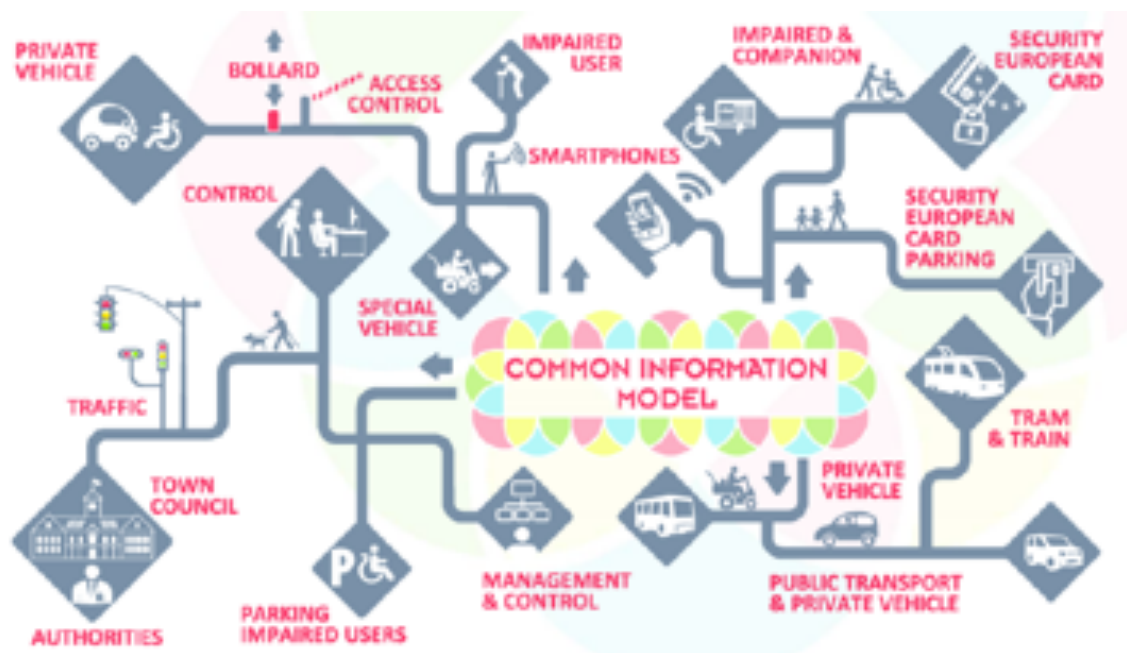


Figure 2.16: SIMON Project Context

The project has as first objective to implement fraud prevention with an evolution of the actual European Disabled Badge, through the usage of ICT enhancements, mainly adding a printed User ID, an NFC badge and a QR Code to the actual card, and providing the usage of virtual access tokens, through mobile devices. The second objective is to provide multimodal navigation, indoors and outdoors, to disabled and elderly people, maintaining user’s privacy, through the usage and interoperation of already existing open data, formatted in a common information model.



Figure 2.17: ICT enhanced European Disabled Badge proposed by SIMON project

Regarding the proposed architecture and modules for the SIMON project it is proposed in a separate public deliverable [20], and can be resumed in the following form:

There are three main applications:

- SIMON LEADS which is the end users' application that allows them, through NFC, to use the enhanced cards, and provides a interface with the services SIMON ANSWERS, SIMON BOOKS, SIMON OPENS and SIMON SAYS.
- SIMON CONTROLS used by the authorities, which implements an interface with SIMON BACKOFFICE, allow for the validation of the parking spot usage by the end users, checking the badges NFC or QR Code, and submitting the results back to SIMON BACKOFFICE, in case of fraud.
- SIMON BACKOFFICE which is comprised of several data repositories and services for accessing them, like users' authentication and management, access to parking resources and registration of incidents like fraud.
-

Regarding the 4 main Services:

- SIMON ANSWERS is responsible for providing SIMON LEADS with the navigational guidance, indoors and outdoors, and providing useful information like elevators and parking spots location.

- SIMON BOOKS give SIMON LEADS the information about parking spots availability and allow SIMON LEADS to make reservations of the available spots.
- SIMONS OPENS allows to SIMON LEADS to open barriers and gives access to reserved parking or restricted areas.
- SIMON SAYS deals with all the necessary authentication and validation of the users.

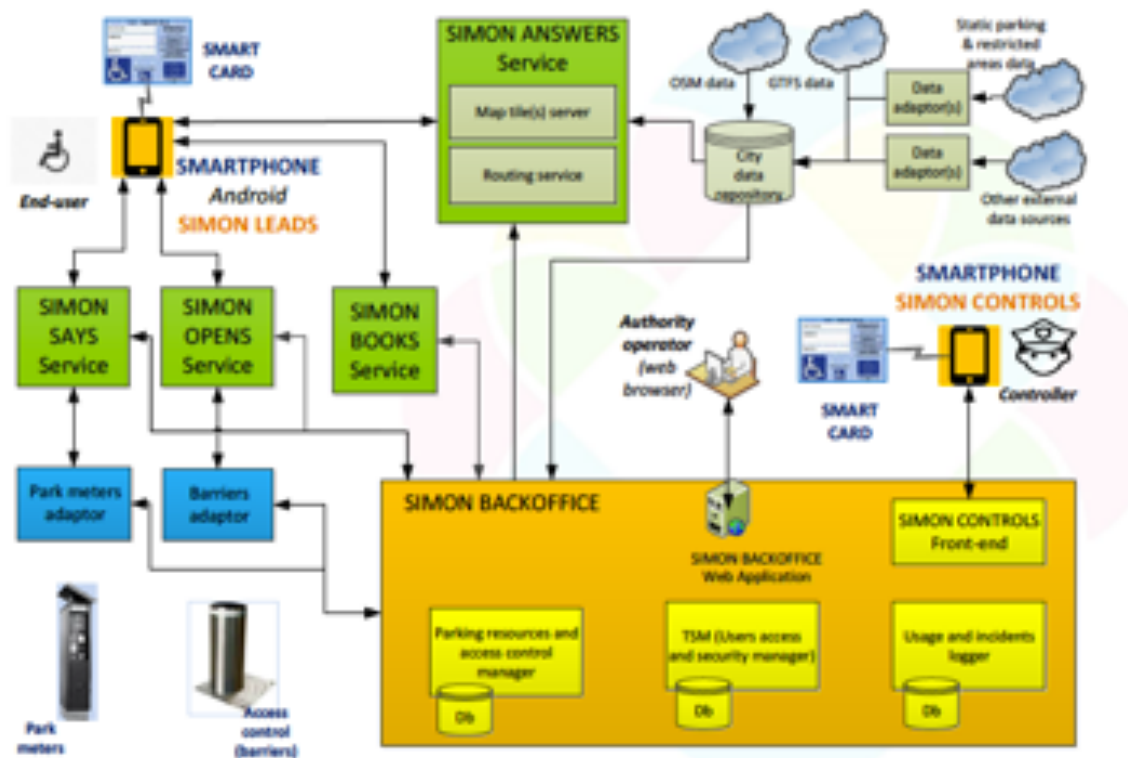


Figure 2.18: The proposed architecture for SIMON project

In Portugal, the SIMON project management was attributed to EMEL, the company that already manages the mobility and parking in the city of Lisbon, being responsible for the implementation, execution, evaluation, communication and dissemination of the project in all its phases. There are four deliverables already available: a webpage with the georeferencing of all the available disabled parking spaces in the city of Lisbon, a channel dedicated to people with reduced mobility so they can expose the daily obstacles and problems, a pavement repainting program, to further dissuade abusive parking, and the last one a reinforcement of parking officers, and their inspection of fraudulent situations.

Finally, the commercial solution. And why this commercial solution and not another one? Because this was the first contact with reserved parking validation, as it is implemented in one of the reserved parking spaces of a local mall.

The service is called phonepark [21] and it was developed by two Portuguese companies: EST and Epocavia. The first implementation was in Forum Montijo shopping mall, where the system was deployed in four parking slots, where two are reserved for disabled, and the other two for pregnant women, elderly people or people with babies.



Figure 2.19: Phone Park reserved parking spots in Forum Montijo shopping mall.

How does the system work: each parking slot as an unique toll free phone number assigned to it. By making a call to this number, the user is either given access to the parking slot, in case it is registered and has access to that certain parking space or receives an SMS message with the conditions to access that parking space.

In case the user has access to the spot, the barrier lowers, and the user can park the car. Immediately after the user leaves, the barrier lifts again, preventing abuse parking.



Figure 2.20: Parking space with access barrier and information panel.

2.4 – Conclusions

There many varied solutions based in several technologies, and more will appear as the IoT and Smart Cities concepts evolve. More and more these solutions are becoming modular and using Cloud resources, so it can become easier to rollout an implementation and then scale when the demands arise. Most of them are nowadays sending feedback to remote users, mainly with the usage of mobile devices and applications, also allowing in some cases the remote reservation, and guidance to the reserved parking space, making use of traffic information and navigation apps.

There are a large number of possibilities regarding sensors and devices that allow for the identification of cars and help to understand the occupancy of parking spaces. Some of them are better in some contexts, like closed parking, other work better in open environments.

The IoT paradigm helped to evolve the proposals, not only because of the diversity of devices that are nowadays interconnected, but also the proliferation of small computing devices like Arduino or Raspberry Pi, that bring great computing power, for these embed applications, in a very small footprint with a reduced price.

Also, the new communication protocols being proposed thinking mainly in IoT applications, like 802.11ah and 802.15.4 [22], are enlarging the capabilities of these devices,

increasing their communication range, while reducing interference and lowering consumption requirements.

But after analyzing all the proposals, it can be concluded that the majority of them focus on closed parking lots, instead of providing solutions for street parking, and only a few of them focus on disabled parking or can be adapted to serve the validation of parking in these reserved spaces.

From here, the objectives for the next chapters of the dissertation, are to grab and make use of all this knowledge presented in this chapter, and build a proposal, mainly focused in the disabled parking validation, offering the best flexibility and usability made possible by all these technologies and concepts.

3

Chapter 3 – Solution's Architecture

In this chapter a solution is presented, and its architecture devised, giving some alternatives for some areas, and answers why some technology was chosen for a particular area, or why some area of architecture is done in the chosen way.

After all the learnings and vast choice of sensor technologies and communication stacks, referred in the last chapter, there are some immediate takings:

- The system should be modular, scalable and able to address multiple situations and requirements without great changes or to reimplement everything from scratch. Things like being able to adapt to different parking positions, and be able to deal with street parking, which is where the most infractions occur;
- Should take advantage of all the Cloud Services being offered these days. It's a faster way to jump to implementation phase, and also a great way to start the project without making huge investments in management and information system areas. Also, there are already Cloud Services specially designed for IoT, like AWS IoT, that get you a jump start in implementing the solution, have high security standards, implementing advanced features like AI and machine learning from the inset, and making use of the huge AWS partners network, to increase the proposal's value [23];
- The application of WSNs is another great taking, not only because of the proliferation and advances of these devices, but also as good measure for energy consumption performance and networking capabilities. Nowadays a small board like ESP32, packs enough processing power with a lot of available GPIOs, in a very small package. It also has embedded Wi-Fi, Bluetooth and BLE communication

capabilities, and allows for add-ons through an extremely huge range of sensors and peripherals. Its several energy saving schemes, industrial reliability and extreme working temperature ranges, make a perfect candidate for WSN implementation in IoT scenarios [24];

- Regarding the communication stacks, in addition to BLE, there are other proposals that take in consideration all the basic requirements of IoT implementations. From the more common usage of ZigBee, which nowadays as become a *de facto* technology in IoT projects, to newer protocols like LoRa, SigFox, NB-IoT and others, there is a great diversity, and huge choice range, with great benefits in performance – LoRa is a wide area protocol that reaches 15 km in range – without sacrificing features like battery powered supplies and data security;

Technology	Band	Speed	Range	Energy Consumption	Implementation Costs	Dessimination of Devices	Security
4G	Cellular (from 700MHz to 2600Mhz)	Max 300Mbps	150km Max	Very High	High	Very High	High
Wi-Fi	The more common are 900MHz, 2.4GHz and 5GHz	802.11ax: 9608 Mbps Max 802.11ac: 6933 Mbps Max	Between 100 meters and several kilometers depending on antennas	Medium	Low	Very High	Medium to High
Bluetooth/BLE	2.4GHz	3Mbps Max	100m Max	Low/Very Low	Low	Very High	Medium to High
ZigBee	2.4GHz	250kbps Max	100m Max	Low	Medium	Medium	High
LoRa	169MHz, 433MHz, 868MHz and 915 MHz	50kbps Max	15km Max	Low	Medium	Low	Medium to High
SigFox	433MHz, 868MHz and 915MHz	100bps Max	50km Max	Low	Medium	Low	Low
NB-IoT (Narrow Band)	Cellular (from 700MHz to 2600Mhz)	250kbps Max	15km Max	Low	High	Low	High
LTE-M	Cellular (from 700MHz to 2600Mhz)	1Mbps Max	11km Max	Low	High	Low	High
Wi-Fi HaLow	900MHz	347Mbps Max	unknown	Low	Low	In Development	Medium to High

Table 3.1: Comparison of Communication Technologies for IoT [22], [25], [26], [27], [28]

- Another important knowledge is the need to have reservation services available to the users. With this service, the users can have the assurance that the parking spot is available for them, at the reserved time spot, without having the possibility, of when reaching the

parking spot, this is already taken, and the user needs to roam again to find another available spot;

- Systems and applications with a huge degree of usability and availability are another requirement. The mobile phone penetration, in Europe, in 2017, has surpassed 85% of the population with almost 500 million people using the mobile services [29], so everyone has access to a mobile device nowadays. And people are getting more and more accustomed to using it in the daily routine. So why not take advantage of this situation and provide a better service with a better chance of being used;
-

Next an architecture draft is proposed, to relate the several modules of the implementation, and introduce how the system works.

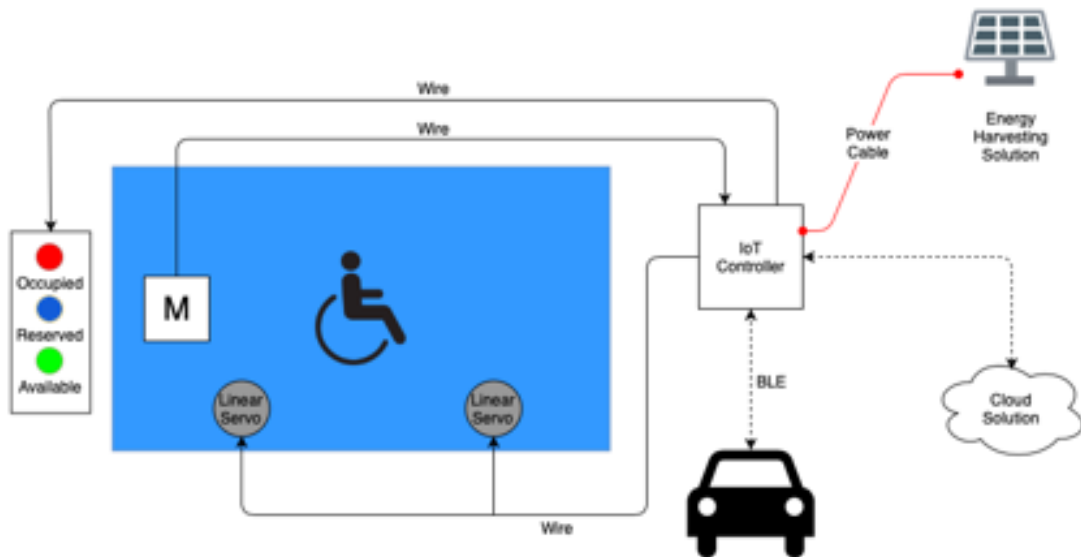


Figure.3.1: Proposed Solution Block Diagram.

First, a local visual feedback is considered, so that any user, or authority agent, can immediately understand the status of the parking spot. This is achieved through the usage of three colored lights: red for occupied, blue for reserved and green for available.

Then the parking spot, with a magnetic sensor, to output the state of the spot's occupancy.

The user can see the available parking status nearby him, through a interface, that shows each parking space status, and availability. Also, in this interface the user can make a reservation for a parking space that is available.

Then, when the user comes nearby the parking space, the Node reads the unique identifier being shared by the user's mobile phone, by the usage of BLE, and opens the barriers to allow for the user to park.

3.1 – Communication stack choice

The de facto communication protocol/technology for the last few years, was ZigBee. But things have evolved, and the limitations of ZigBee are already showing, mainly its limited coverage range of around 100 meters.

With the introduction of 5G, the IoT and Smart Cities concepts, a need for more performance, lower energy consumption and better range arise, which introduced several new protocols, grouped under the LPWAN proposals. These protocols although using lower bitrates (which is not an disadvantage in IoT environments where the size of the information to share is most of the times small), have much bigger ranges in the size of kilometers, and present energy consumption so small, that most of the batteries for these systems can reach 5 or more years of life.

The two finalist choices for the communication stack, are NB-IoT and LoRa. They are the best choices for a solution like the proposed here, at the time of writing.



Figure.3.2: LoRa and NB-IoT comparison[27]


	Advantages	Disadvantages
	QoS implementation	Dependent of Telco Operators which implies a monthly Telco Invoice
	Higher data rate and lower latency	Uses licensed spectrum that costs more than 500 million dollars per MHz
	It uses already deployed 4G network	Regular sync consumes more battery

Figure.3.3 – NB-IoT advantages and disadvantages


	Advantages	Disadvantages
	Asynchronous protocol allow better battery performance	Initial cost of implementation
	Greater coverage range per gateway	Lower data rate and higher latency
	100 to 1000 dollars each gateway	
	Uses unlicensed spectrum which is cost free	

Figure.3.4 – LoRa advantages and disadvantages

What made the choice needle move to LoRa? The fact that LoRa, although less cost effective at the implementation phase, will be almost free to run, during the project's timeline, while NB-IoT presents a monthly expense, that needs to be paid to the telco operators. The fact that several projects propose to deploy an open LoRa network in the major world cities, allowing everyone to use shared gateways, supports even further this decision.

3.2 – Hardware solution choice

When it comes to choose which hardware, would support the solution, and taking in consideration the already chosen LoRa stack, there were several options to take in consideration.

The first one, the safest and easiest path to an implementation, was to choose any Arduino platform (either Uno or even Nano) and then use one of the already available HATs that would provide the LoRa communication.

But some further research allowed to understand that the computation power needed for a solution like this one is not huge, and an Uno would be overkill for this implementation – event taking in consideration the low price of an Uno nowadays. Also having a smaller footprint and some sort of integration with LoRa would be a benefit.

With these two new considerations in mind, the research led to two final candidates: the Pycom LoPy4 and the Adafruit Feather 32u4 LoRa Radio.

The Pycom LoPy4 board is a development board, four bearer (Wi-Fi, Bluetooth, LoRa and SigFox), based on the Expressif chipset, with a footprint of 55mm by 20 mm.

It can deal with simultaneous connectivity of SigFox and LoRa, presents very low energy consumption compared with other microcontrollers, and its supported on a very well-known programming language, Micro Python,

From the product's datasheet[30] here are the specifications:

3.0 Specifications

3.1 CPU

- Xtensa® dual-core 32-bit LX6 microprocessor(s), up to 600 DMIPS
- Hardware floating point acceleration
- Python multi-threading
- An extra ULP-coprocessor that can monitor GPIOs, the ADC channels and control most of the internal peripherals during deep-sleep mode while only consuming ~25uA.

3.2 Memory

- RAM: 520KB + 4MB
- External flash: 8MB

3.3 WiFi

- 802.11b/g/n 16mbps

3.4 Bluetooth

- Low energy and classic

3.5 RTC

- Running at 150kHz

3.6 Security

- SSL/TLS support
- WPA Enterprise security

3.7 Hash / encryption

- SHA
- MD5
- DES
- AES

Figure.3.5: PyCom LoPy 4 specifications

and its pinout:

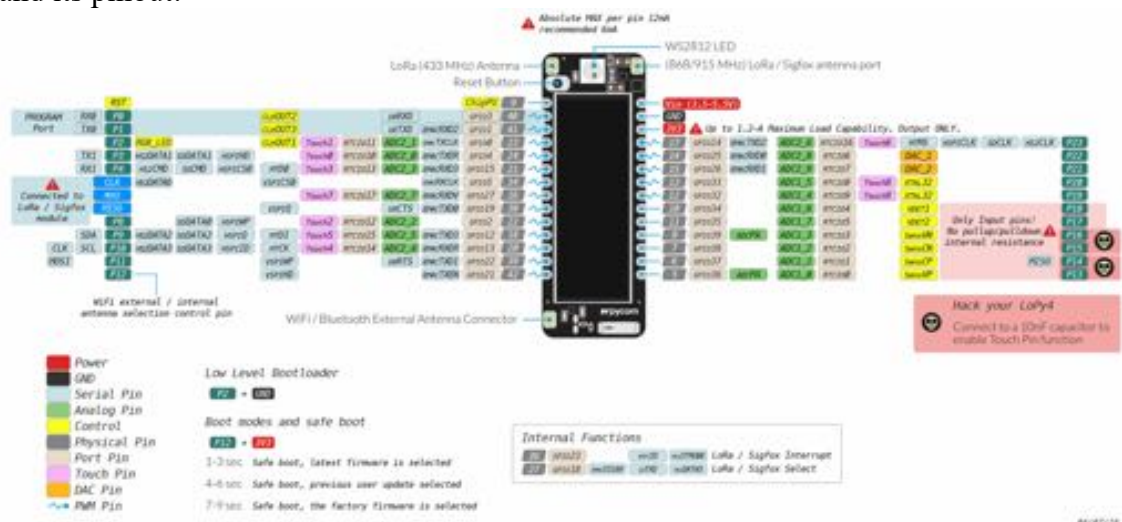


Figure.3.6: PyCom LoPy 4 pinout

Regarding the Adafruit Feather product, it's also a very small footprint development board, based on ATmega32u4 microprocessor, with 32K of flash and 2K of RAM, with built in USB-to-Serial program and debug capabilities. Makes use of packet radio with available Arduino libraries, so its ramp-up development time is very short.

From the product's website [31] here are the product's main specifications and advantages:

- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather - 5.5 grams
- ATmega32u4 @ 8MHz with 3.3V logic/power
- 3.3V regulator with 500mA peak current output
- USB native support, comes with USB bootloader and serial port debugging
- You also get tons of pins - 20 GPIO pins
- Hardware Serial, hardware I2C, hardware SPI support
- 7 x PWM pins
- 10 x analog inputs
- Built in 100mA lipo charger with charging status indicator LED
- Pin #13 red LED for general purpose blinking
- Power/enable pin
- 4 mounting holes
- Reset button

Figure.3.7: Adafruit feather 32u4 main specs

and its pinout:

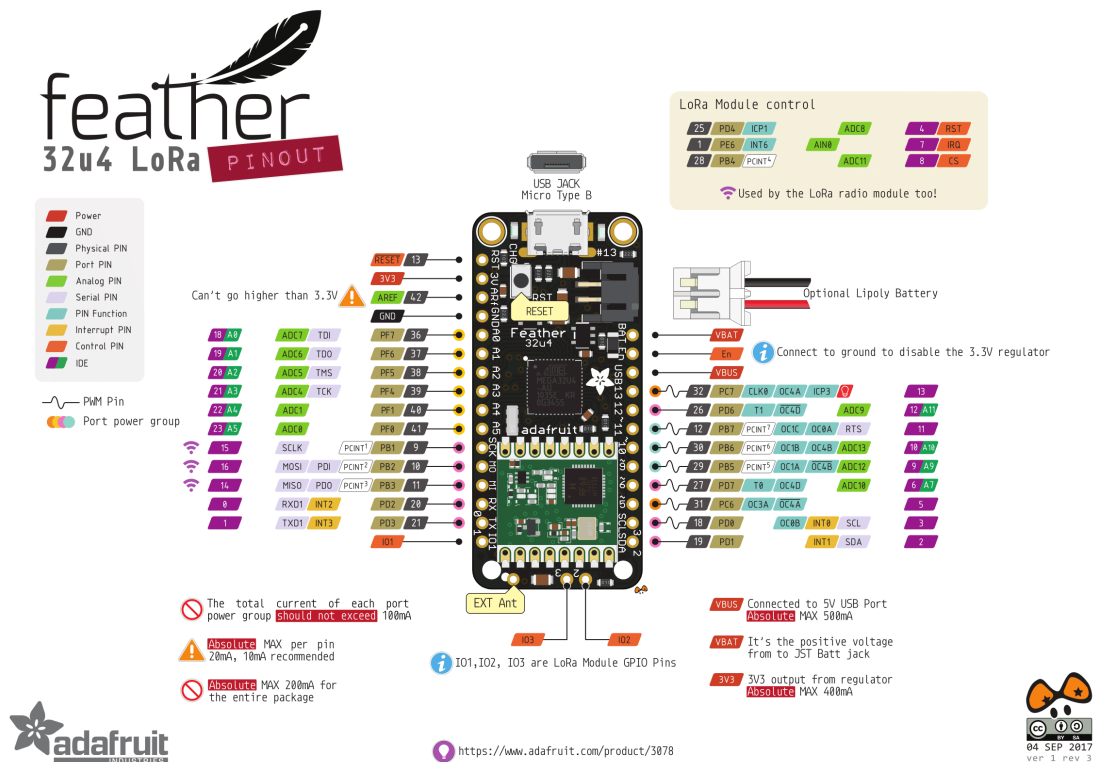


Figure.3.8: Adafruit feather 32u4 LoRa pinout

The final decision was up to one of Pycom LoPy 4 biggest advantages, the fact that it is capable to serve as a Nano LoRa gateway, which is ideal for a development scenario like the one in this dissertation, dispensing with the need to buy a dedicated LoRa router/gateway, for which the cheapest, single channel, available model cost north of one hundred euros, and some multichannel models costing between two hundred and six hundred euros.[32]

The total cost for two Pycom LoPy 4 boards (34,95 euros each at the time of the thesis writing), with dedicated LoRa/SigFox antennas and a Expansion Board 3.0 (to make code prototyping and USB uploading to the device easier), was around one hundred euros.

This allows for a parking node module (sensors, LEDs, motors and other components will be added later) and a central Lora Nano gateway that can, in theory cover a 4 to 10 kilometers radio area, serving as a gateway for several parking nodes in this radial area.



Chapter 4 – Solution's Implementation

4.1 – Methodology

The approach to this part of the project was to define a strategy to tackle all the various parts and components of the project, trying to reduce road-blocks e moving forward fast. Starting small and simple, a Proof-of-Concept (PoC) idealization, and then, by adding features, but still keeping focus on simplicity of the product, and its prototyping, building it up to reach the final prototype.

This not only allows for a faster and more straightforward way of moving forward with the project, it's also less risky, because at the end of each step, a working prototype is available, and if for some particular reason, after a certain step, there was no more time to develop to project, it would still be possible to present something that works, instead of having some disconnect components, not able to communicate and give a general idea of the project.

For every step of the project, documentation needs to be done, with the goal of having a full picture when finished, and the capability to reproduce every step, if needed.

4.2 – IDE choice

The Integrated Development Environment (IDE) of choice for this project, was Atom. It was an easy one, because it was already being used for some professional and home-made projects, either running Python or Golang.

Atom is very lightweight and simple in its usage, and the fact that most of these projects are so lightweight in size and number of files, there is no need to move to more advanced IDEs.

Atom also handles dependencies very efficiently reducing worries when compiling or running the projects.

The only specific software needed in addition to the Atom was to install Pymakr Plugin for Atom following the instruction available at PyCom website [33].

The usage of a Mac computer meant that there was no further need to install drivers for the hardware, as this hardware was automatically recognized by the operating system.

4.3 – Communication Middleware definition and setup

Now that the hardware for either the Node and the Gateway was chosen, the first thing to understand and decide at this step, was how LoRaWan works and if there was already some platform or libraries available to achieve the support for this implementation.

The first step was to reach an understanding the definitions of LoRa and LoRaWan, and what distinguishes them.

LoRa is the definition of a wireless communication technology, patented from 2013 and 2008, based on spread spectrum radio modulation. It was originally developed by a French company, by the name of Cycleo, and it was bought as a promising wireless technology by Semtech.

Its main advantages are the large reach in distance, trading it for the bitrate/bandwidth available, that is mainly oriented to small data interchange, instead of sending bandwidth consuming protocols and services like audio and video.

The maximum bandwidth attainable is around the 11kbps when the Nodes are relatively close to the gateway, while the reach can, at least, theoretically, reach the 30-kilometer range in line-of-sight, while keeping the energy consumption to a minimum.

The LoRaWan is a MAC protocol, that allows Nodes to communicate with the Gateway in bidirectional and multicast fashion, that allows for great reliability on the communication, with increased levels of security. It's already a standard that allows for the interoperability of networks in different regions of the globe. Another great feature of LoRaWan, is the capability of registering Nodes, through the usage of Over-The-Air mode.

There are already a few solutions in the market, to perform the role of communication middleware, with at least two, showing simplicity in the integration, and easiness of usage: Loriot and The Things Network (TTN).

Regarding the Loriot solution, it's being marketed, as three different products:

- Network Server, which handles all the network management, and also the OSS and BSS layers;

- Application Server, taking care of device and data access management;
- Join Server, that allows secure management and storage of keys;

Loriot.io infrastructure for Internet of Things

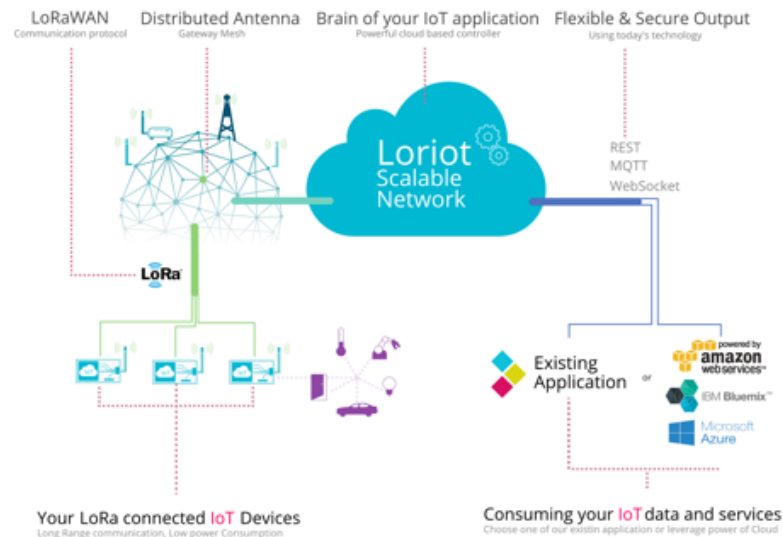


Figure.4.1 – Loriot Infrastructure for IoT

The product applicable to this project, is the Network Server, in the shared instance with community account model, which is free, for a single gateway, and up to 10 Nodes, that allows for the evaluation of the services, to build PoCs, and the usage in small scale, non-critical production environments.

The Network Server main features are:

LORIoT NETWORK SERVER	
Role	IoT Connectivity Middleware
Purpose	Network management, operations (OSS) & billing (BSS)
Function	LoRaWAN to IP / IPv6 translation Network Management & Operation Data Streaming Key, security & privacy management
Focus	Network-centric
Typical customer	LoRaWAN Network Operator City / municipality Utility Alternative operator Existing M2M / IoT providers
Pricing	Capacity based (per gateway / device)

Figure.4.2 – Lorient Network Server Features

The only disadvantage of this product is the requirement to use one of their approved solutions. As the usage of the Pycom Lopy4 as gateway was already taken, and the fact that at time of writing this device was not supported as a Gateway solution, automatically made this platform be rejected.

The next platform analyzed was The Things Network (TTN). TTN leads a more open-source approach to the LoRaWAN ecosystem, trying to help each and every individual or enterprise, interested in bringing a solution to the market.

The approach is divided in four phases:

- Explore: TTN allows any individual to plunge into their knowledge base, and gain knowledge from the very first steps, right up to very advanced configurations and feature of the ecosystem. They also give access to very simple configuration tools that allow for very fast and intuitive setup;
- Proof of Concept: allows for the definition of the business model, analyze all the different use-cases and decide which model (build / buy) you're going to use for the hardware development;
- Prepare to scale: at this stage it's time for looking at essential subjects like security and scaling of the solution and defining the level of QoS;
- Run in production: running the infrastructure with full confidence and being able to make updates on the run are essential subjects at this phase;

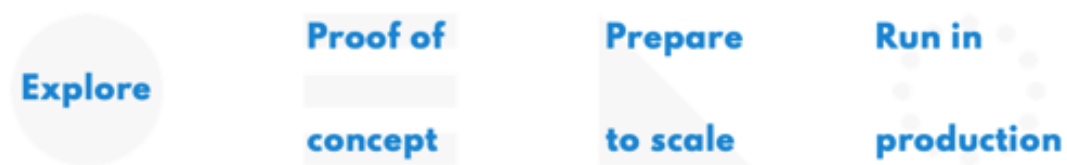


Figure.4.3 – The LoRaWan Project approach proposed by TTN

Regarding the much needed support in technological projects, TTN offers three levels of support depending on the type of client: the free tier, where a forum with knowledgeable members and professional serves as the base for asking doubts and getting help; the professional tier, where a dedicated professional or a team of professional (depending on the project size) can guide in taking decisions and development of the product; the last tier which is partners channel, allows for a full support, because these are companies that are already deploying and selling products over the TTN network.

Based on all the support and documentation available for everything LoRa and devices related, the option was to join the TTN network, by creating an account and start building the project on top of its services.

4.4 – Setting up the hardware

For easiness of setup and convenience, it was decided to use a PyCom EXP board 3.0. This daughter board allow for easy setup and adds a much needed microUSB interface, which is fundamental for quick prototyping and testing of software versions.

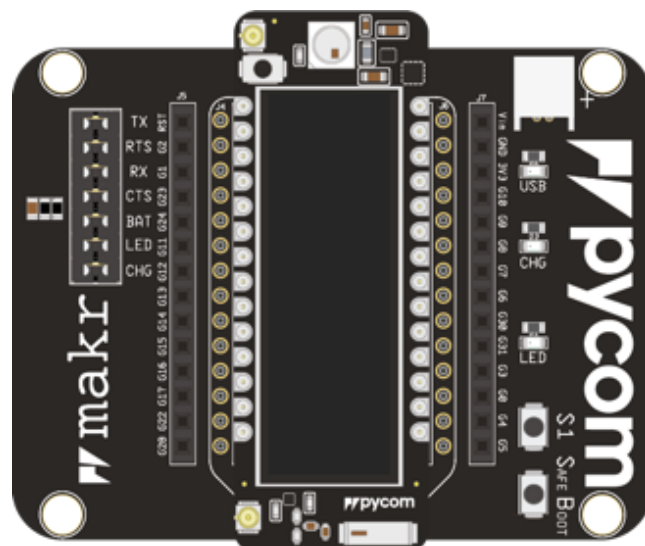


Figure.4.4 – Pycom EXP board 3.0 hardware

Next, and as instructed in the PyCom website, as the LoPy4 module can't work without an antenna, there is the need to plug a approved antenna – in this case the PyCom indoor antenna with pigtail cable:



Figure.4.5 – LoPy4 with the antenna connected

4.5 – Setting up the Nano-gateway

Making use of the detailed instructions available at the PyCom website, regarding their LoPy4 hardware usage as a Nano-gateway [32], the first step was to register the Gateway on the TTN Console, by choosing Gateway option inside the console:

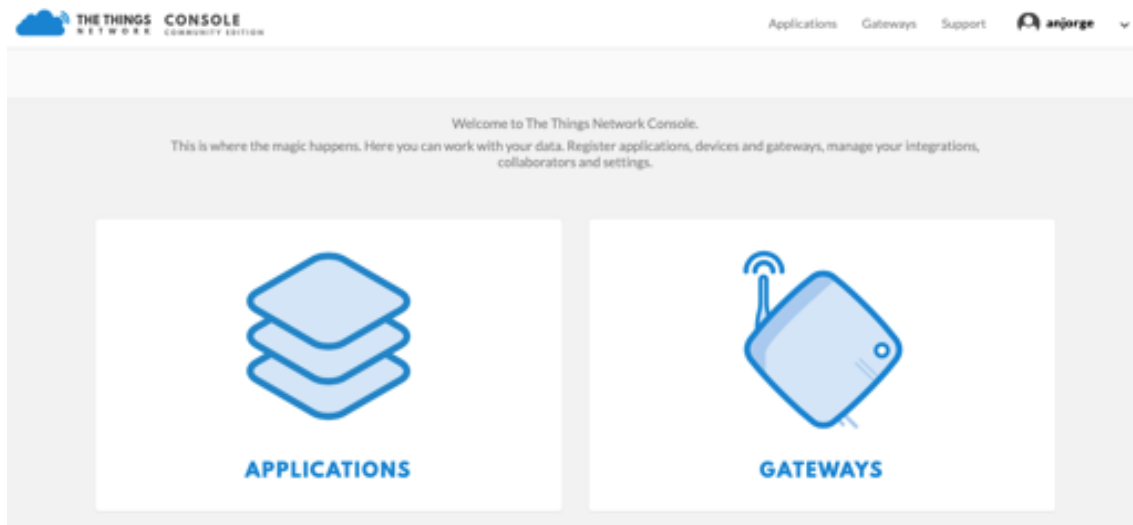


Figure.4.6 – The TTN console

And then just fill the data presented in the next page's form:

Gateway
Register

REGISTER GATEWAY

Gateway ID
A unique, human-readable identifier for your gateway. It can be anything as creative as you like!


☐ **I'm using the legacy packet forwarder**
Select this if you are using the legacy [Semtech packet forwarder](#).

Description
A human-readable description of the gateway.

Frequency Plan
The [frequency plan](#) this gateway will use.

Router
The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.

Location
The exact location of your gateway. This will be used if your gateway cannot determine its location by itself. Set a location by clicking on the map.



Lat: 0.00000000
Long: 0.00000000

Antenna Placement
The placement of the gateway antenna.

☐ indoor ☐ outdoor

Cancel


Register Gateway

You are the network. Let's build this thing together. → [The Things Network](#)

Figure.4.7 – Gateway registration input form

The first field is the Gateway ID, that is user generated. A valid procedure is to generate this ID from the Wi-Fi MAC address (this way the generated Gateway will be simple to reproduce, if needed, and unique since it's based of the Wi-Fi MAC address).

I've used this code to generate the ID:



```
config.py | gatewayid.py | main.py
1 from network import WLAN
2 import ubinascii
3 wl = WLAN()
4 ubinascii.hexlify(wl.mac())[:6] + 'FFFE' + ubinascii.hexlify(wl.mac())[6:]
5
```

Figure.4.8 – Gateway ID generator

Then there is the need to choose the *“I’m using the legacy packet forwarder”* option.

Next option, **Description** is just a label for human readability.

The **Frequency Plan** is related to the frequency plan allowed in the region of the world the Gateway is deployed – in this case, Europe, so the Frequency Plan is **Europe 868MHz**. Also, the next option, which is **Router**, is also related with the location, and the option is **ttn-router-eu**, which is the closest, to reduce latency in the communications between the Gateway and TTN’s router.

Next, for a Gateway without global positioning feature, there is the need to input its **Location** in a Map, so the system can report its Latitude and Longitude coordinates.

Last option is related to the type of antenna, mainly if it’s an outdoor or indoor antenna, which in this case is **Indoor**.

After clicking in **Register Gateway** button, the interface presents a confirmation page, with all the details regarding the configured Gateway:

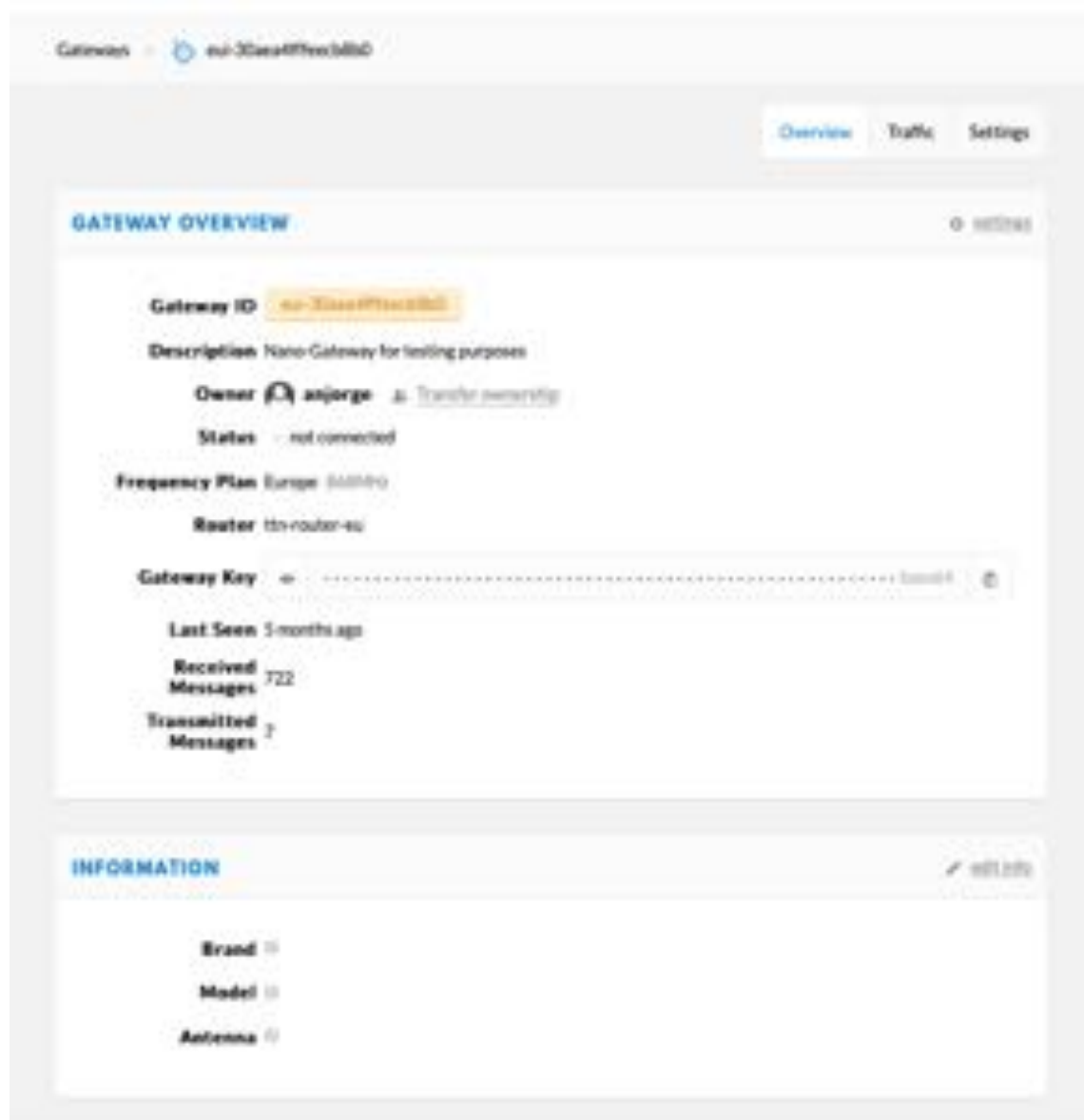


Figure.4.9 – Gateway details after registration – first half of screen

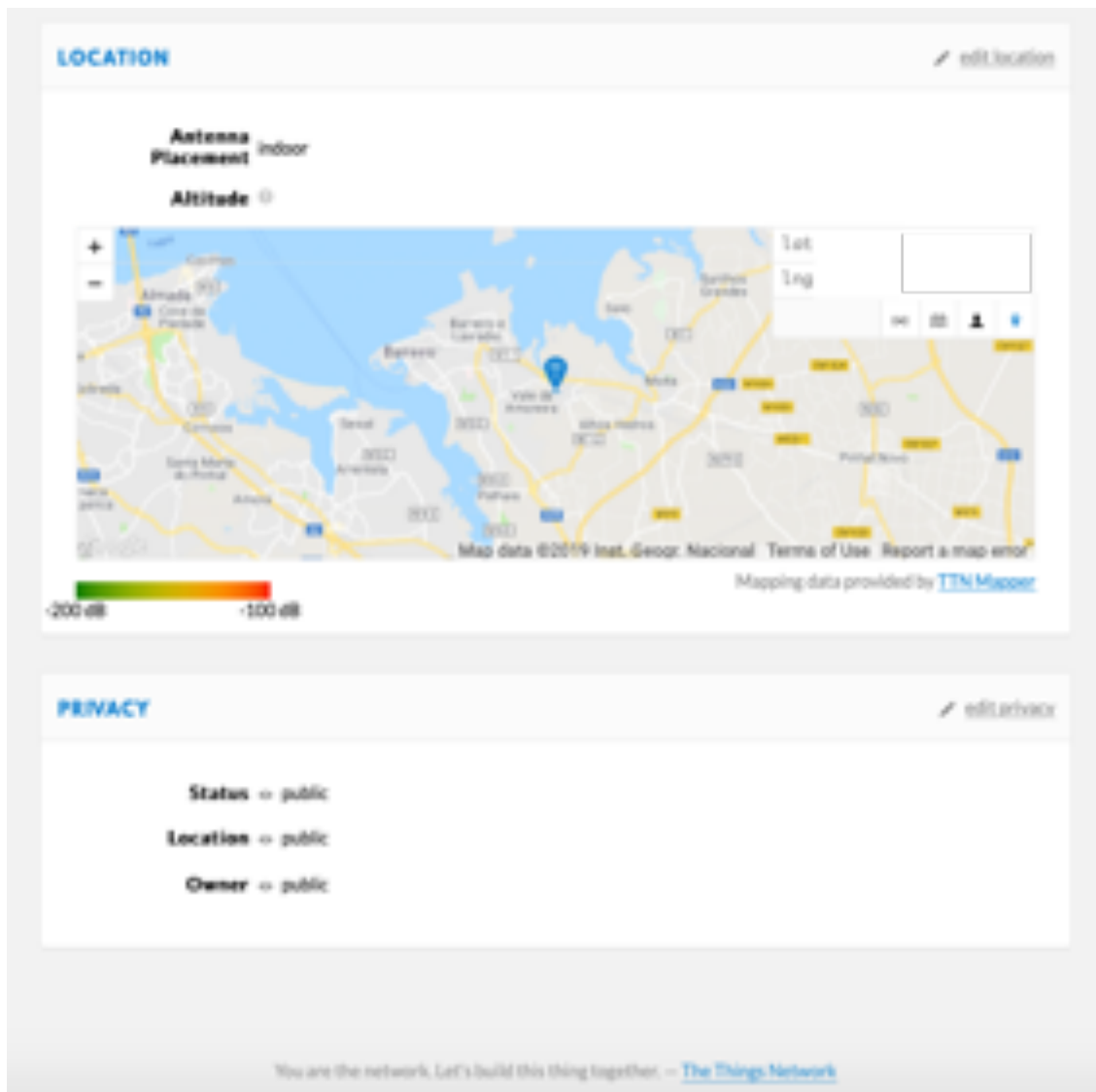


Figure.4.10 – Gateway details after registration – second half of screen

Now it's time to setup the Gateway by uploading the software to the LoPy4. There are three software files:

- config.py which handles all the needed parameters for the Wi-Fi and LoRa communications;
- main.py which runs at startup, making use of config.py to read the parameters and afterwards calling the nano-gateway library;
- nanogateway.py which is the packet handler, responsible for sending and receiving all the packets to and from the TTN server;

Regarding the config.py file, most of the work is already done. Next step is to input the Wi-Fi credentials (WIFI_SSID and WIFI_PASS variables), make sure the correct lines for the frequency (either **EU868** or **US915**) are (not) commented, and that the most “nearby” TTN server is the one configured (**router.eu.thethings.network** in this case).

Neither the `main.py` nor the `nanogateway.py` files need input from the user. The only thing the user needs to make sure is that the `nanogateway.py` file is the most up-to-date version available at the PyCom GitHub repository.

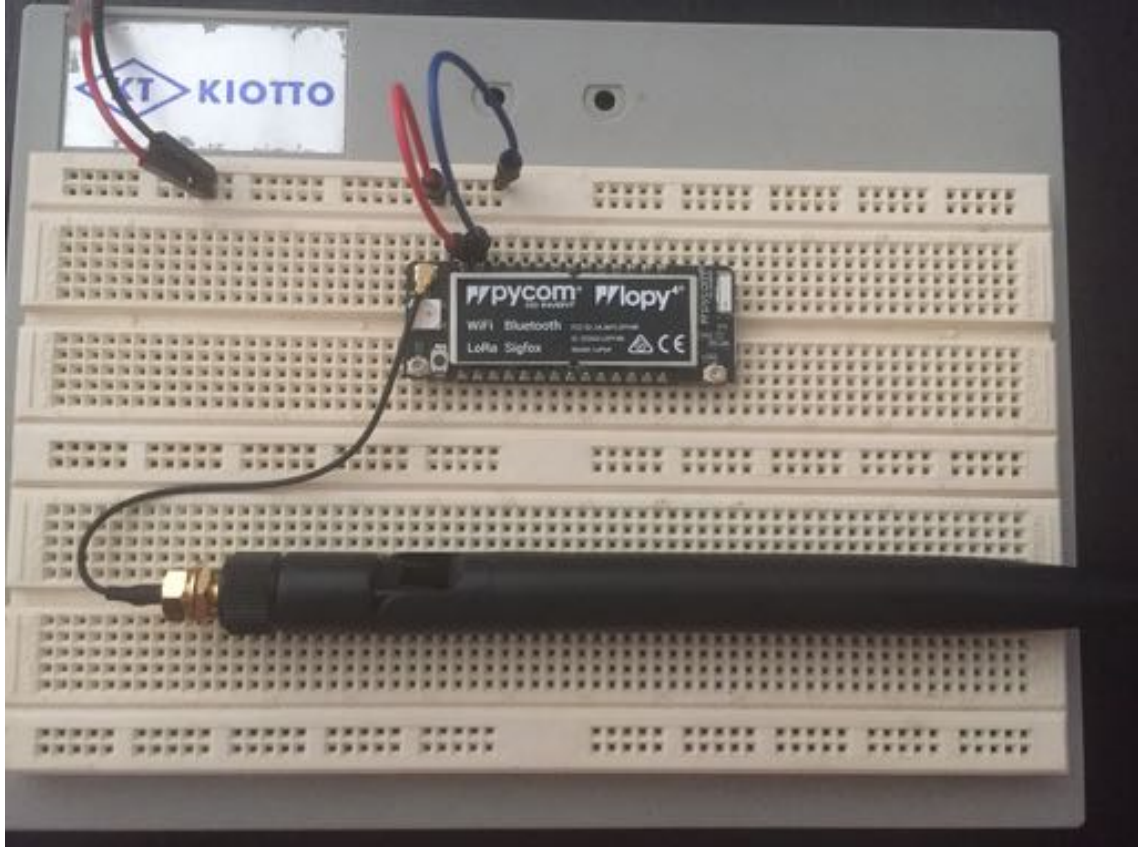


Figure.4.11 – Gateway up and running on a breadboard



Figure.4.12 – Gateway connected and exchanging messages with TTN

4.6 – Simple Node to Gateway Communication

After reading lots of information about LoRaWan, and how one establishes device connection [34] using TTN and a gateway.

First step is to create an Application on the TTN console: on the console, select *Applications* and then choose *add application* button.

 The screenshot shows the 'ADD APPLICATION' form with the following fields:

- Application ID:** A text input field for the unique identifier.
- Description:** A text input field with a placeholder 'Eg. My sensor network application' and a green checkmark.
- Application EUI:** A text input field with a placeholder 'EUI issued by The Things Network'.
- Handler registration:** A dropdown menu with 'ttn-handler-eu' selected and a green checkmark.

 At the bottom right, there are 'Cancel' and 'Add application' buttons.

Figure.4.13 – Adding a new application on TTN console

After filling the previous form, the following information is presented:

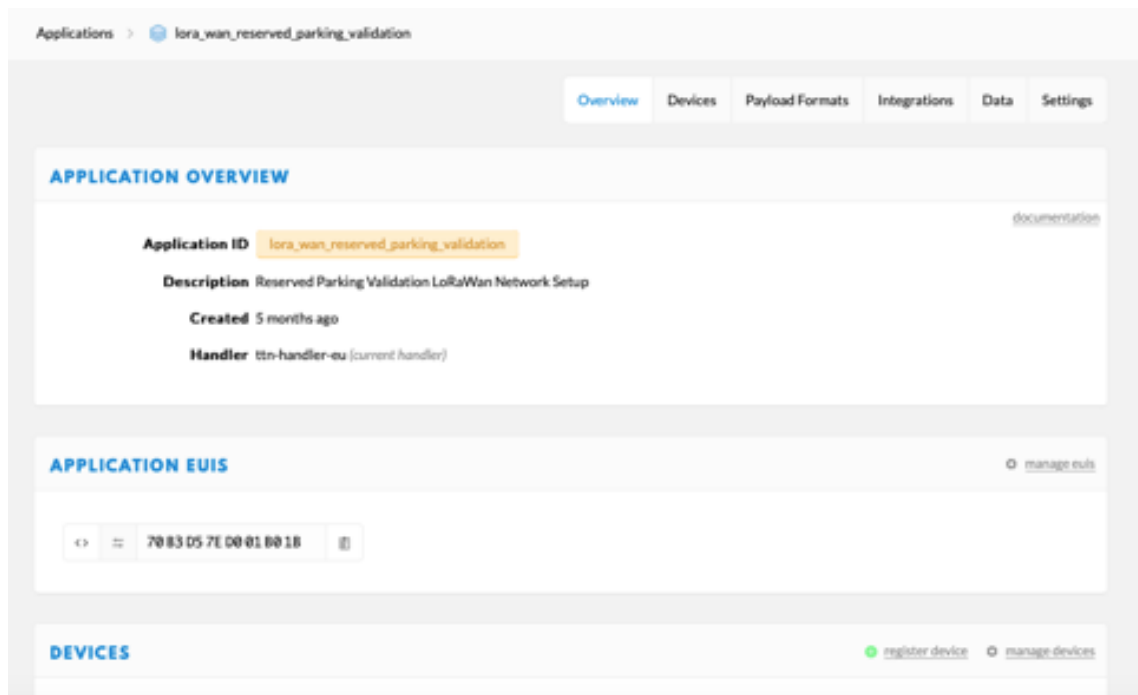


Figure.4.14 – Application Overview and EUI

Next, on the devices option, the first device needs to be registered, which is the Node. This is accomplished by pressing *register device* button:

The screenshot shows the 'Applications' page with the breadcrumb 'lora_wan_reserved_parking_validation' and the 'Devices' tab selected. The 'REGISTER DEVICE' form is displayed, with a 'bulk insert devices' link in the top right corner. The form contains the following fields and instructions:

- Device ID:** This is the unique identifier for the device in this app. The device ID will be immutable. (Empty text input field)
- Device EUI:** The device EUI is the unique identifier for this device on the network. You can change the EUI later. (Text input field with a '0x' prefix and '0 bytes' indicator)
- App Key:** The App Key will be used to secure the communication between you device and the network. (Text input field with a 'this field will be generated' placeholder and a copy icon)
- App EUI:** (Text input field containing the value '7083057E00010010' and a copy icon)

At the bottom right of the form, there are 'Cancel' and 'Register' buttons.

Figure.4.15 – Registering a new device for this Application

The DeviceID needs to be immutable, as it is a unique identifier for this device. The DeviceEUI is the unique identifier in this network, and it can be either inserted by the user or generated by the console. Also generated automatically by the console, will be the App Key.

There are two methods of connecting a Node to a Gateway:

- ABP – Authentication by personalization: In this case the keys given on the Application setup are manually configured on the Node, and with this method there is no need to have a first handshake between Node and Gateway, and the node can immediately send packages to the Gateway;
- OTAA – Over The Air Authentication: the Node only has knowledge of the APPEUI and the APPKEY, and the first step is an handshake between the Node and the Gateway, where the Node makes a join request to the gateway, sending this APPEUI and APPKEY information to the gateway. If the gateway accepts the keys, gives a join confirmation to the Node, who can now send packets do the Gateway;

The code for the two methods can be found at the annexes.

Regarding the main.py file that is uploaded to the Node LoPy, there are 3 main blocks:

```
3 from network import LoRa
4 import socket
5 import ubinascii
6 import struct
7 import time
8
9 # Initialize LoRa in LoRaWAN mode.
10 lora = LoRa(mode=LoRa.LORAWAN)
11
12 # create an OTA authentication params
13 dev_eui = ubinascii.unhexlify('000836793678B5E8') # these settings can be found from TTN
14 app_eui = ubinascii.unhexlify('7003D57ED001B01B') # these settings can be found from TTN
15 app_key = ubinascii.unhexlify('A6E8D3C6FF3C5B46CA21D43BFAC9C603') # these settings can be found from TTN
16
17 # set the 3 default channels to the same frequency (must be before sending the OTAA join request)
18 lora.add_channel(0, frequency=868100000, dr_min=0, dr_max=5)
19 lora.add_channel(1, frequency=868100000, dr_min=0, dr_max=5)
20 lora.add_channel(2, frequency=868100000, dr_min=0, dr_max=5)
```

Figure.4.16 – Initialization block

On the Initialization block, all the external libraries are imported, the module is initialized in LoRaWan mode, the Dev EUI, the App EUI and the App Key are set, as the activation is set to OTAA, and finally the setup of three channels on the same frequency, for the communication with the Gateway.

```

25
26 # join a network using OTAA
27 lora.join(activation=LoRa.OTAA, auth=(dev_eui, app_eui, app_key), timeout=0)
28
29 # wait until the module has joined the network
30 while not lora.has_joined():
31     time.sleep(2.5)
32     print('Not joined yet...')
33
34 print('Joined')
35
36 # remove all the non-default channels
37 for i in range(3, 16):
38     lora.remove_channel(i)
39
40 # create a LoRa socket
41 s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
42
43 # set the LoRaWAN data rate
44 s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
45
46 # make the socket non-blocking
47 s.setblocking(False)
48
49 time.sleep(5.0)

```

Figure.4.17 – Join block

On the Join block, a *lora.join* command is issued to the Gateway, setting the mode as OTAA and passing the three variables (Dev EUI, the App EUI and the App Key) set on the Initialization block.

Then it waits for a *lora.has_joined* confirmation from the Gateway, and when joined prints message *Joined* on debug console.

Next, removes all the non-default channels (3 to 16), creates a LoRa socket, setting the LoRaWan data rate, and setting the socket to non-blocking.

```

47 for i in range (200):
48     s.send(b'PKT #' + bytes([i]))
49     time.sleep(4)
50     rx = s.recv(256)
51     if rx:
52         print(rx)
53     time.sleep(6)

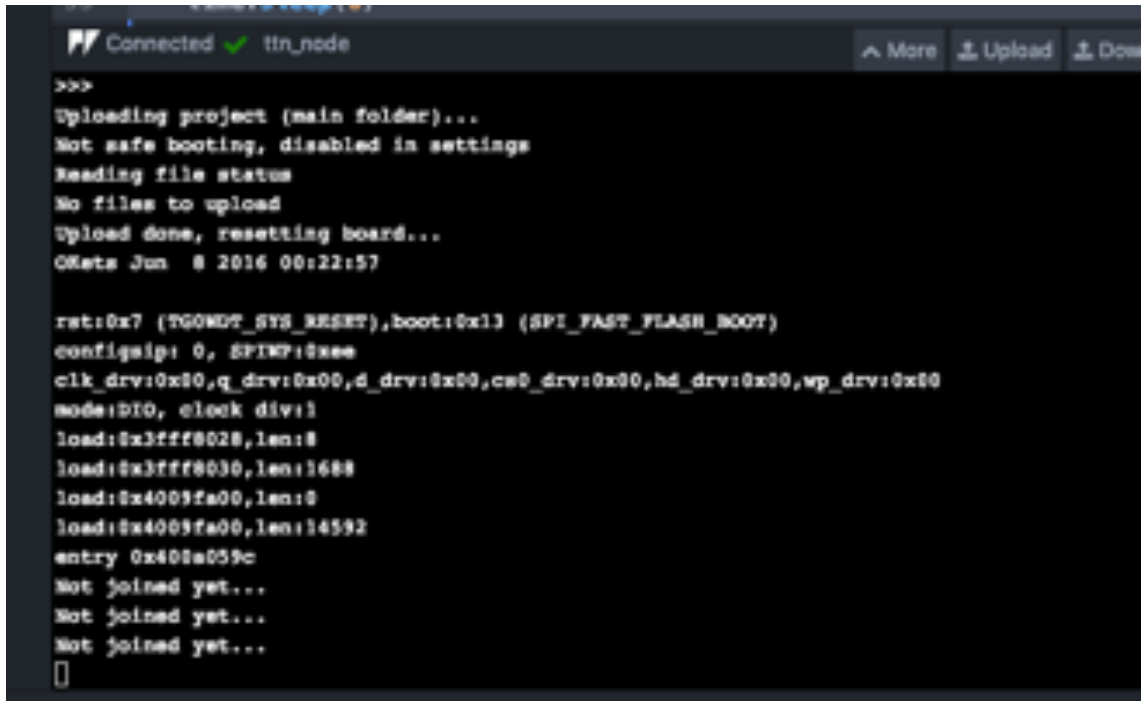
```

Figure.4.18 – Test block

On the Test block, at this phase, what is accomplished is to send packets to the Gateway and listen to packets sent from the Gateway to test the connection.



Figure.4.19 – Node in debug connected through micro USB port



```
>>>
Uploading project (main folder)...
Not safe booting, disabled in settings
Reading file status
No files to upload
Upload done, resetting board...
OMeta Jun  8 2016 00:22:57

rat:0x7 (TGOWDT_SYS_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3ffff8028,len:8
load:0x3ffff8030,len:1688
load:0x4009fa00,len:0
load:0x4009fa00,len:14592
entry 0x400a059c
Not joined yet...
Not joined yet...
Not joined yet...
[]
```

Figure.4.20 – Upload of Node software and debug messages during the Join phase



Figure.4.21 – Activation package on the communication initialization on TTN console



Figure.4.22 –Communication flow (upload) and the respective payload

DOWNLINK

Scheduling

FFPort

Payload

0 bytes

SIMULATE UPLINK

FFPort

Payload

0 bytes

Figure.4.23 – On the TTN console sending data to the Node or to the Gateway can be simulated

14:34:36
0
1
payload: CEAADA

Uplink

Payload

CEAADA

Fields

no fields

Metadata

{
"time": "2019-09-17T13:34:36.714756143Z"
}

Figure.4.24 – Check the receive on the Gateway

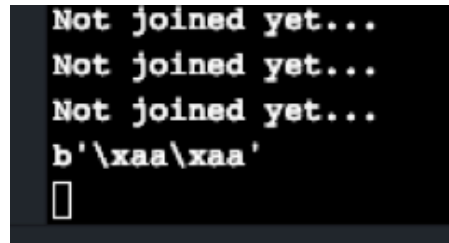


Figure.4.25 – Check the receive on the Node (through debugging console on Atom)

4.7 – Reading sensors on the Node

The magnetic sensor that was chosen to correctly identify the presence of a car in the reserved parking space is based on the Honeywell HMC5883L[35], which is a surface mount chip, capable of low field sensing of magnetic fields, and that presents a good cost to performance ratio. The chip provides a I²C serial interface, giving a resolution of up to 5 milligauss output over 3 axis, and with a very low current usage,

The pinout of the board is:

- GND;
- VCC – accepting values between 2.16-3.6VDC;
- SDA – Serial Data Line;
- SCL – Serial Clock Line;

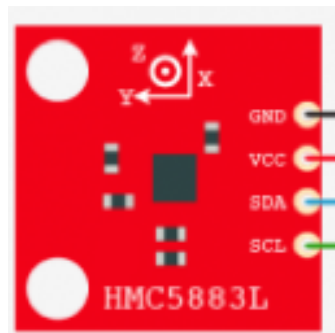


Figure.4.26 – Honeywell HMC5883L pinout (I²C)



Figure.4.27 – Sensor connections to the LoPy 4

After the reading of information about this sensor and after the performance of some tests with I²C read code (available in the annexes), no output values different from zero, were obtained from the sensor, when trying to read changes. After some more research, the conclusion that the sensor was always outputting zero, because it was not calibrated, was reached.

After a while, a library usable with the LoPy, that allowed to read the values from the three axis, without the need to properly calibrate the sensor, was found. This would not present a problem, because in this project, the main goal, was not to reach a precise tridimensional orientation, but to read a certain change, resulting from a big metal object (a car) moving or parking over the sensor.

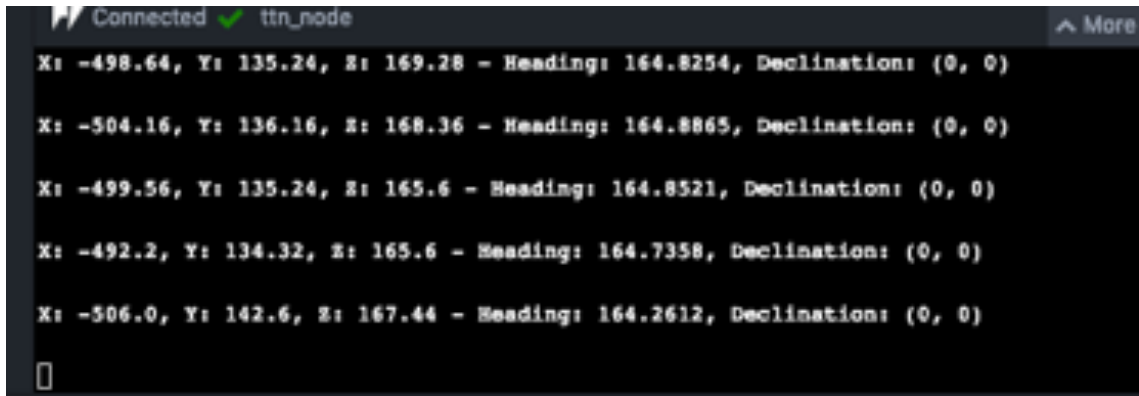


Figure.4.28 – Positioning readings from the sensor – three axis, heading and declination

With this library, and adapting the code already written, the strategy was to read the starting position at the beginning of the Test block and set those values as the definitive “location” values.

From here, performing some tests, was needed, to understand the minimum threshold, from which to consider that a car was overhead the sensor. Also needed, was to make sure that, at least two readings within a certain interval, were over this threshold, to remove the occurrence of false positives.

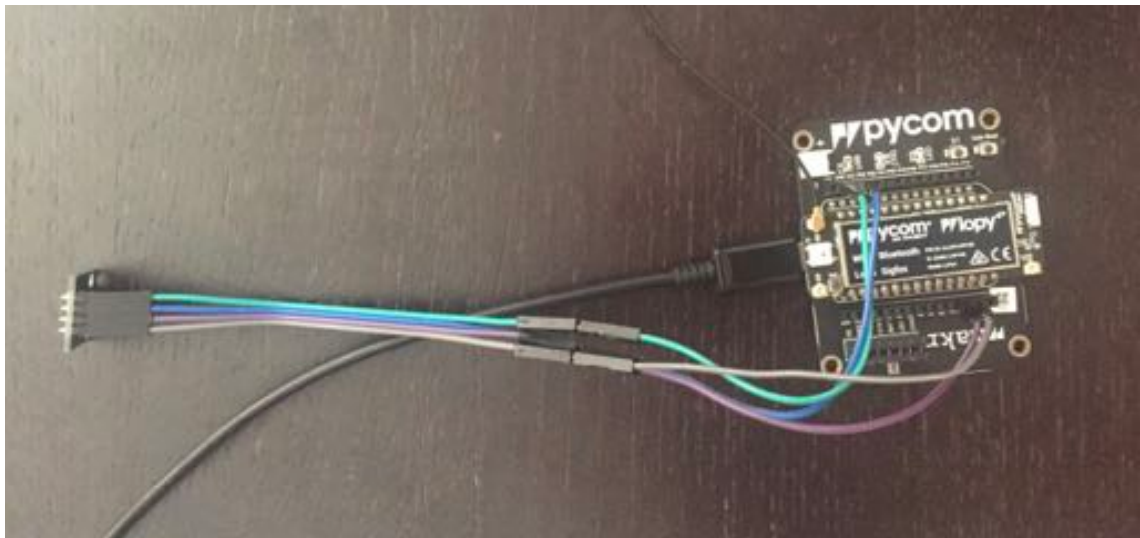


Figure.4.29 – Node with sensor

4.8 – State machine and sending sensor information to the Gateway

The state machine is the main component of the software running on the Node. It is this component that rules all the decisions regarding the fact that the parking space is available, occupied or reserved.

From these three states the state machine is built, and the conditions are:

- If the Node receives a message that the parking space is reserved and the parking space is not yet occupied, the Node acknowledges this fact and turns the state to reserved;
- If the sensor states that the parking space is occupied, the Node acknowledges this fact, and turns the state to occupied;
- Otherwise, or in case the Node receives a message that the parking space is no longer reserved (withdrawal) - the parking space state is available;

As the Nodes are remote systems, that might have failure/absence in communication, and to persist the state of the parking space, every time the state changes, a message is sent to the Gateway, for further processing.

The messaging to the Gateway is in very simple format, and makes usage of the send function over the socket already established:

- If the parking space is occupied, the Node sends the byte 0x01 to the Gateway;
- If the parking space is not occupied, the Node sends the byte 0x00 to the Gateway;
- If the parking space is reserved, the Node sends the byte 0x02 to the Gateway;

```
80 #state machine for parking
81 if reserved == b'\xff':
82     s.send(bytes([0x02])) # parking reserved
83     pycom.rgbled(0x0000FF)
84 else:
85     if (abs(actualX - initialX) > 20) or (abs(actualY - initialY) > 20) or (abs(actualZ - initialZ) > 20):
86         s.send(bytes([0x01])) # parking occupied
87         pycom.rgbled(0xFF0000)
88     else:
89         s.send(bytes([0x00])) # parking empty
90         pycom.rgbled(0x00FF00)
91     time.sleep(10)
```

Figure.4.30 – State machine code

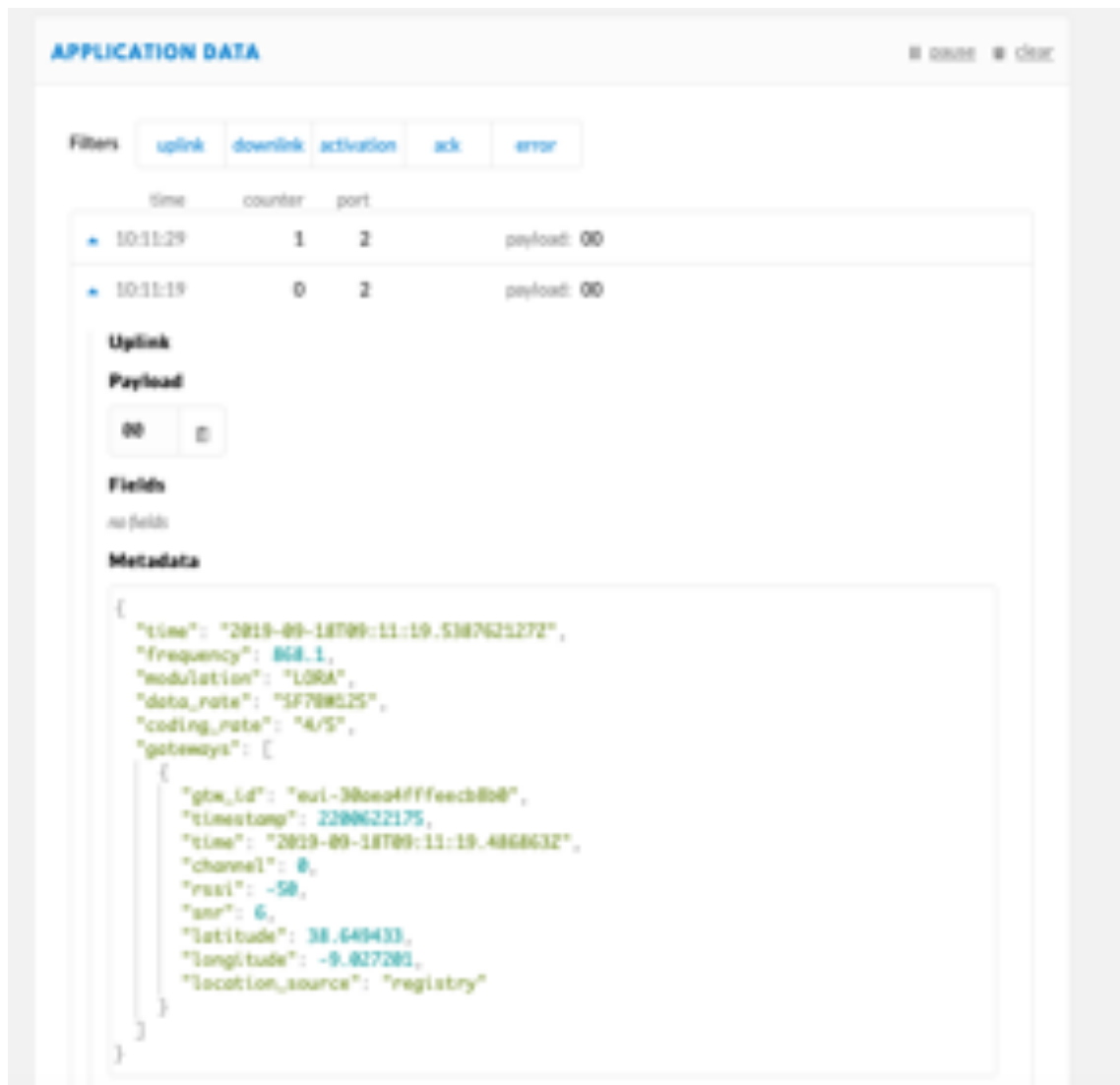


Figure.4.31 – Message sent to the Gateway – not occupied state

One decision that was made, was not to enforce the checking of the status prior to changing the state to reserved. This is to be made at the reservation interface, that only allows to reserve (and therefore send the message to the Node) if the parking space is not already occupied.

4.9 – Introducing local feedback at Node with LEDs

This was a very standard solution as the LoPy4 hardware includes a RGB LED, that is very easily instantiable in the code.

The first step is to include pycom library.

By default, the RGB LED, pulses blue, indicating the heartbeat of the system.

This can be turned off, using instruction:
`pycom.heartbeat(False)`

Then a command can be evoked, that uses 8 bytes for each color in hexadecimal format (R, G and B) to accomplish a range of 16777216 colors: `pycom.rgbled(0xRRGGBB)`

For this project the decision was on three colors:

- Green (0x00FF00) – for when the parking space is free;
- Red (0xFF0000) – for when the parking space is occupied by a vehicle;
- Blue (0x0000FF) – for when the parking space is reserved by an authorized driver;

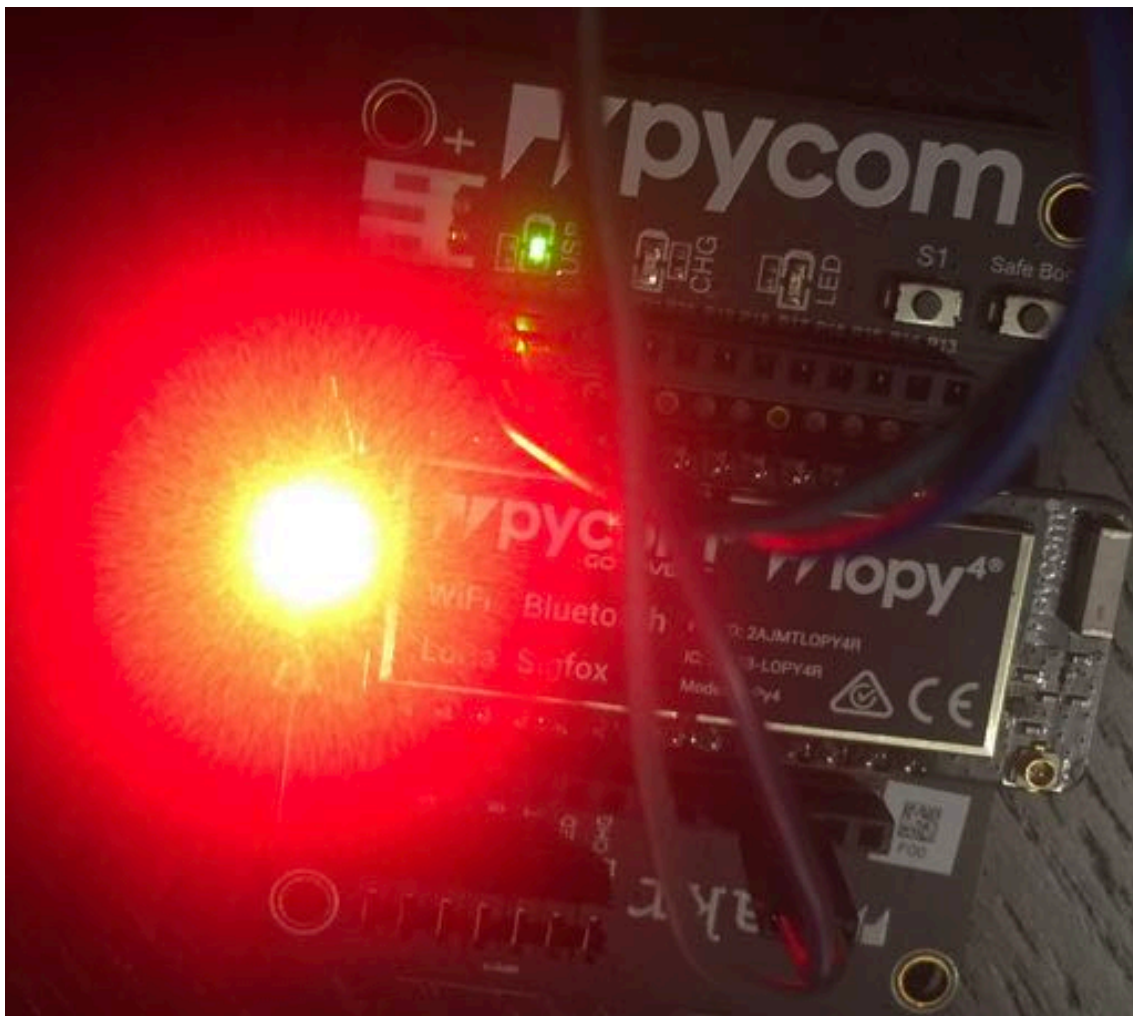


Figure.4.32 – The parking space is now occupied

4.10 – Cloud or not to Cloud? And which one to choose?

Regarding cloud integration, the main aspects that led to this approach and using a cloud provider were:

- Data aggregation: considering the amount of data generated by a system composed of several hundred or maybe thousands of Nodes, relaying their information on the parking spaces availability and status, it makes perfect sense to join all this data in one unique place, where dashboards for customers or management can be easily developed and integrated. Without this aggregation it would be much harder to present a dashboard with all the available parking spaces in a city;
- Scalability and availability: this dissertation only deals with one Node, but if the project would be deployed in a city like Lisbon, with hundred or more disabled parking spaces, the need to manage all these nodes, and guarantee maximum availability for the system, are two situations easily addressable by cloud systems;
- Low starting investment/pay as you go: to move from a PoC like this to a full featured project, always represents a big investment, not only setting the Nodes and Gateway, but deploying physically the needed infrastructure for this kind of service, would require a great setup expense. Using the cloud, reduces this part of the initial investment, allows for a clearer vision of the expenses, and paying just for what you need when you need, instead of a upfront investment in machines, that once again, in 4 or 5 years will be obsolete, another situation that does not occur in the cloud environment (in the end it still occurs in the cloud environments but it's not of users' care anymore);

Of course, there are negatives also, and here are some of them:

- Privacy/security issues: nowadays who is not concerned with security breaches, information and identity thefts, and unauthorized usage of data? Considering that the data present (like license plates, address, and other personal data) is of high sensibility, great cares need to be taken in the security and access layers of the system, to make sure only the ones with authorization can access the data. And a deployment in Europe, would need to address the much talked GDPR compliance[36];

- Communications issues: nowadays nothing works without communications or internet, and this being a project about IoT, it greatly relies on this layer for working. What needs to be addressed in the cloud are the delays, and latency of the communications, and be aware that some of the Nodes, aren't always communicating, and when they communicate those tiny messages, what needs to be guaranteed is that they reach their destination;
- Knowledge/ramp-up: not all the IT workers and specialists of today, are very knowledgeable, or at least have experience in using cloud environments. And the ones that already used these environments, at first, are more oriented to using the cloud environments services like AWS EC2, raising virtual machines, trying to recreate the self-owned or rented Data-center's environments, but this time on the cloud;

The cloud environment chosen was the Amazon AWS cloud services, mainly because of two aspects:

- Familiarity with the environment: at professional level, the author already had some knowledge on the usage of two cloud environments: Amazon AWS and Microsoft Azure. Both have their advantages and disadvantages, and both have tools and services specially dedicated to IoT projects. But nowadays the biggest experience is with Amazon AWS, where in the last 6 months, good knowledge about serverless and stateless services (lambda services) and also non-relational databases (DynamoDB), was gathered;
- Integration with TTN: the support, and all the documentation that TTN offers, about the integration with the Amazon AWS cloud, is also of extreme significance, and allow for much more peace of mind, when starting the integration;

4.11 – Cloud integration

After reading all the documentation, that TTN provides, about the integration of their systems with Amazon AWS, and after creating a free tier account in Amazon AWS, it's now time to use a Cloud Formation package, provided by TTN, that raises automatically the needed Elastic Beanstalk infrastructure/stack on the AWS cloud.

Giving a little bit more depth about these two AWS services/tools, Elastic Beanstalk[37], is a service provided by Amazon, that allows the fast deployment and implementation of a infrastructure/service, without needing to know nor manage all the AWS services behind it. It supports a number of known programming languages, and its manageable by AWS CLI or AWS console.

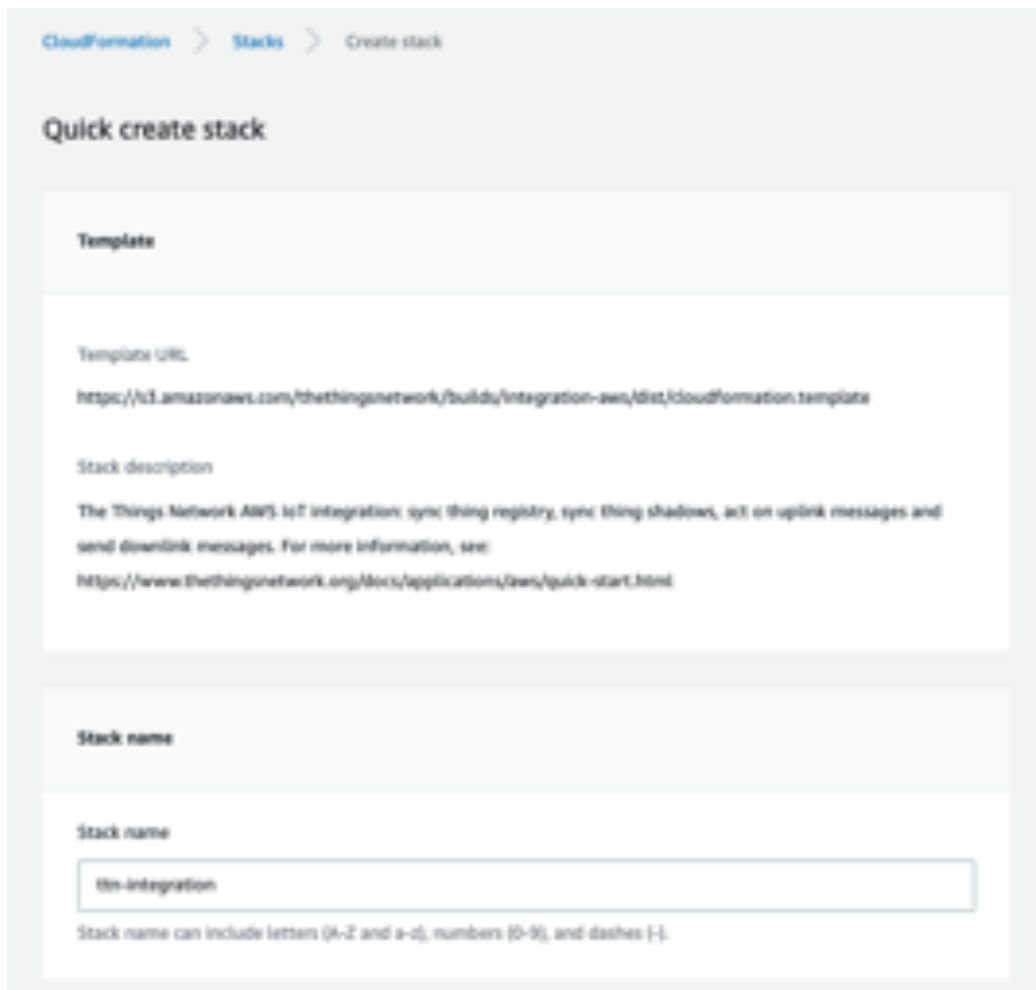
This is also where Cloud Formation[38] comes to the rescue. Introduced as one of AWS tools for the “Infrastructure as Code” paradigm, it allows to create infrastructure models and replicate environments through the usage of code files, based on a single common coding language. Another big advantage of Cloud Formation is that it’s a free service for the customer, that only pays for the resources provisioned by the Cloud Formation file, as they are used.

The conjunction of these two services/tools allows for the easy and replicable deployment of all the needed infrastructure to support the integration between TTN and Amazon AWS.

After logging on the AWS Console, and choosing region eu-west-1 (Ireland) for the deploy, just load a Cloud Formation template that is available in an AWS S3 bucket in the following url:

<https://s3.amazonaws.com/thethingsnetwork/builds/integration-aws/dist/cloudformation.template>

There are a number of fields that require to be filled in a form before the creation of the stack:



CloudFormation > Stacks > Create stack

Quick create stack

Template

Template URL

<https://s3.amazonaws.com/thethingsnetwork/builds/integration-aws/dist/cloudformation.template>

Stack description

The Things Network AWS IoT integration: sync thing registry, sync thing shadows, act on uplink messages and send downlink messages. For more information, see: <https://www.thethingsnetwork.org/docs/applications/aws/quick-start.html>

Stack name

Stack name

ttn-integration

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Figure.4.33 – Stack name input

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

The Things Network Settings

App ID
App ID
lotsa_wan_reserved_parking_validation

App Access Key
App Access Key
ttn-account-v2.QY15bzCMWwCagU3T5Hq44nESeWJTTv5NYL_DPH4

Account Server
Account Server address
https://account.thethingsnetwork.org

Discovery Server
Discovery Server address
discovery.thethings.network:1900

AWS Elastic Beanstalk Settings

Environment Name
Elastic Beanstalk environment name
ttn-integration

Figure.4.34 – Stack parameters

Instance Type
EC2 instance type

t2.micro

SSH Key
Name of an existing EC2 KeyPair to enable SSH access to the instance

ssh_integration

AWS IoT Settings

Enable Syncing Things
Enables AWS IoT things synchronization and thing shadow deltas

true

Sync Interval
Interval for AWS IoT things synchronization

1m

Thing Type Name
AWS IoT thing type name. Requires AWS IoT things synchronization

lorawan

LoRaWAN FPort
LoRaWAN FPort for thing shadow delta downlink messages. Requires AWS IoT things synchronization

1

true
Push metrics to CloudWatch for monitoring

true

Figure.4.35 – Stack parameters continued

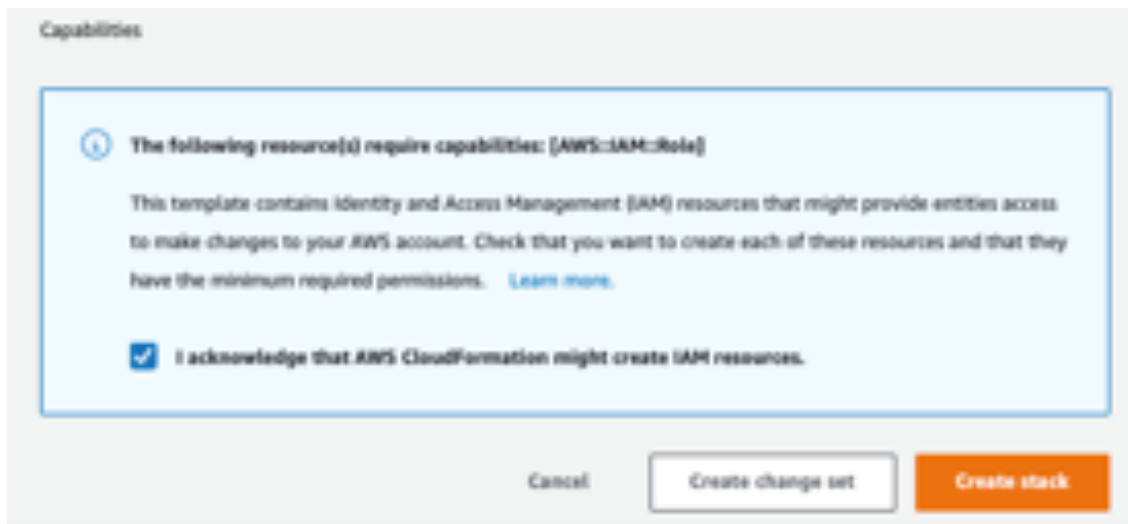


Figure.4.36 – Stack capabilities acknowledge

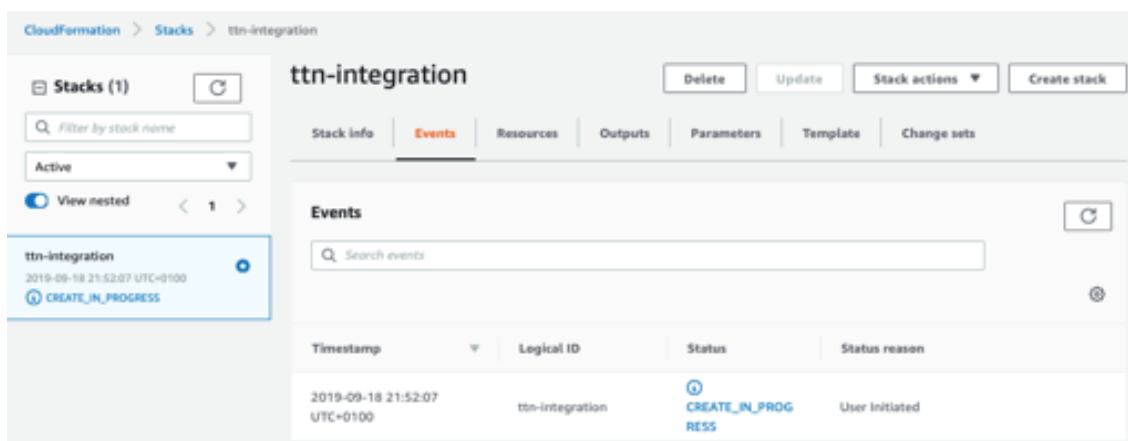


Figure.4.37 – Stack creation in progress

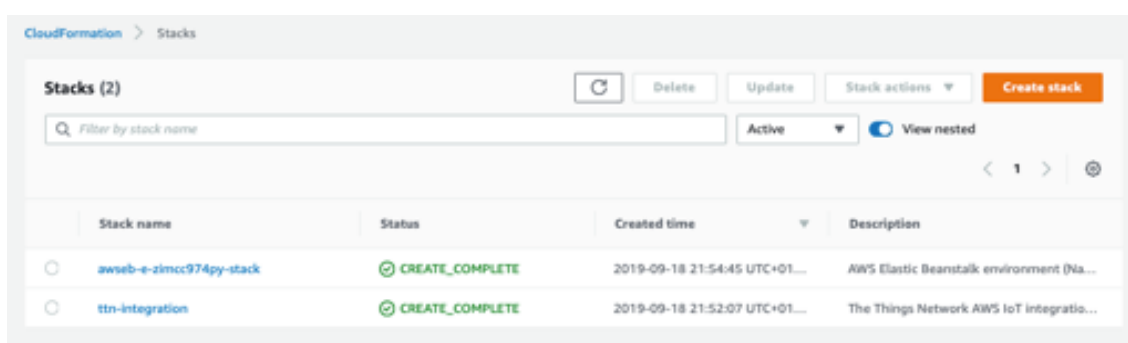


Figure.4.38 – Stack creation complete

The test of the integration, can be achieved at the AWS service AWS IoT, on the console, choosing, on the left menu, the option Manage:

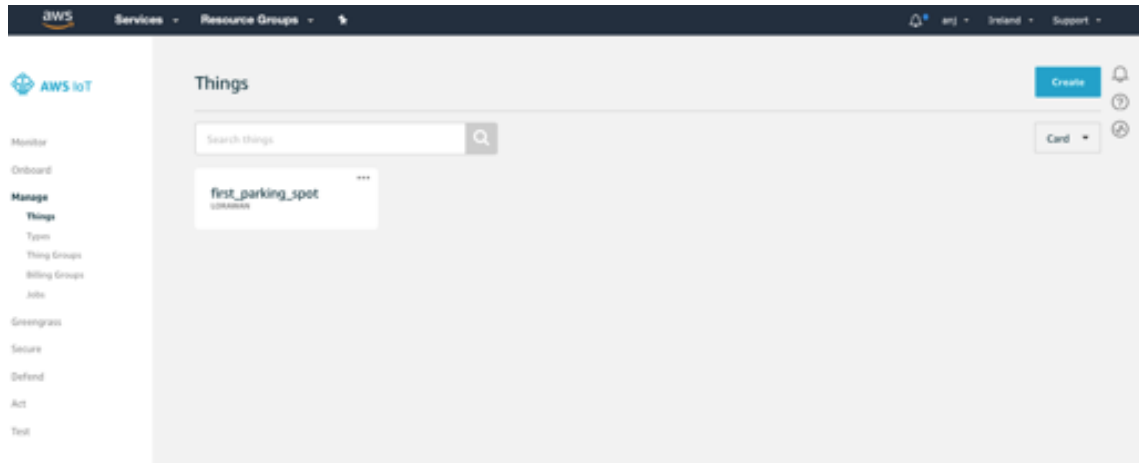


Figure.4.39 – Manage Things on AWS IoT

On the previous image, it can be confirmed that the integration was successful and that the devices (in this case only one) are present and the data is correct.

In this interface it is possible to test sending and receiving messages from the Node to AWS.

Choosing Test on the left menu the following interface is presented:

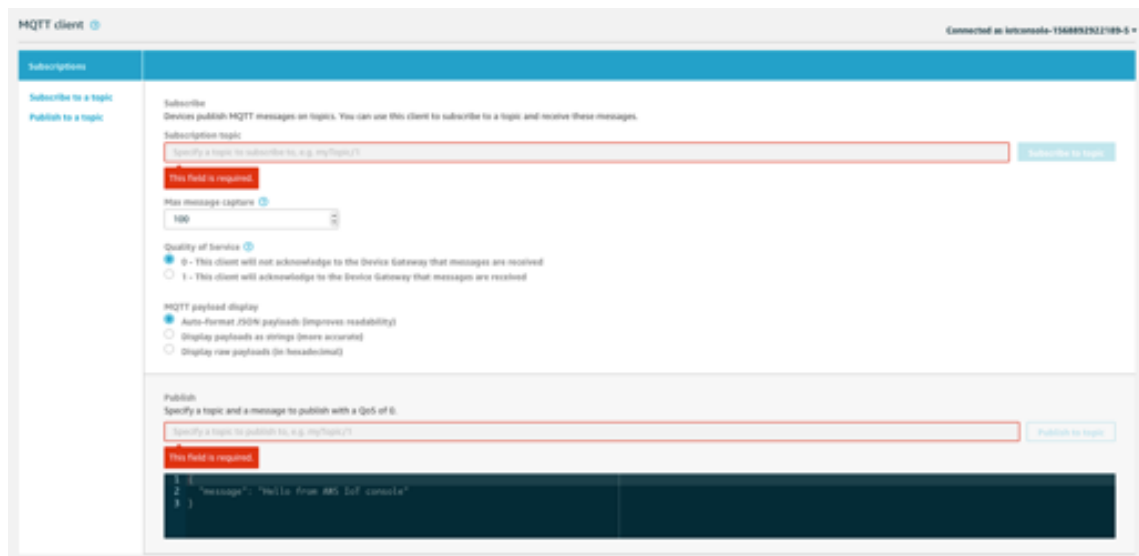


Figure.4.40 – MQTT client for testing on AWS IoT

In this interface it can be either choose to Subscribe to a topic and receive messages from the Node or Publish to a topic to send messages to the Node.

For the Subscribe it needs to be defined which topic to subscribe, in the format: `<AppID>/devices/+/up` (in this case: `lora_wan_reserved_parking_validation/devices/+/up`)

Or to get messages from all the devices from all the apps: `+/devices/+/up`

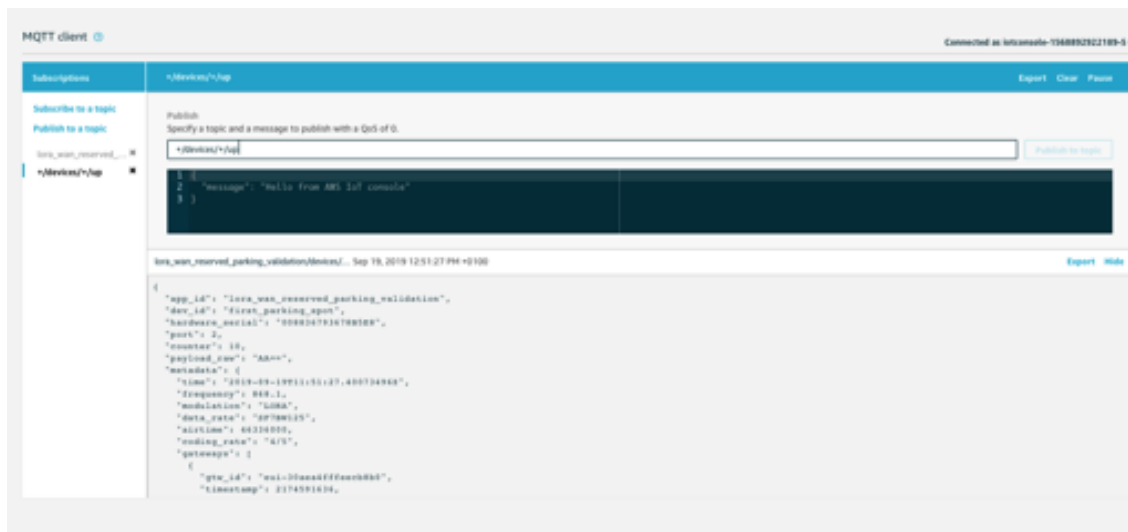


Figure.4.41 – Example of a message sent from the Node (empty parking space)

For the Publish, the topic where to publish the messages, needs to be set, in the format: <AppID>/devices/<DevID>/down (in this case: lora_wan_reserved_parking_validation /devices/first_parking_spot /down).

The message format to send is:

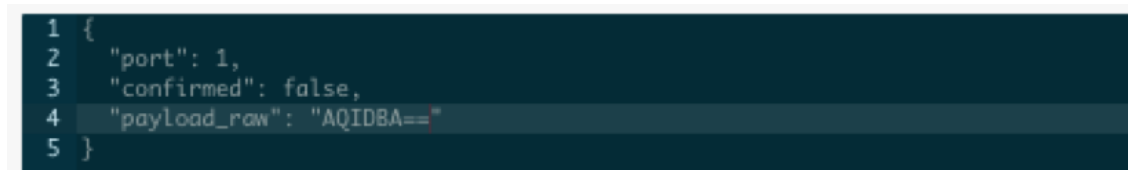


Figure.4.42 – Downlink message format

Before moving to the interface definition, let's get the messages sent from the Node, delivered to DynamoDB for persistence, before sending them to a Lambda function that will run a serverless web site, showing the parking space status.

AWS DynamoDB[39] is a non-relational database, available in AWS cloud, that is document and key-value oriented, and capable of handling huge number of requests, with single-digit millisecond performance.

AWS Lambda function[40] is a service that allows AWS users to run code when needed, and scale accordingly, without provisioning or managing a base architecture. The price is paid based on the usage, and when the service is not executing the user doesn't have to pay for it.

To accomplish this task, first of all, an action, needs to be created, to intercept uplink messages and send them to the DynamoDB. On the left menu the option to choose is Act, and then select Create a rule button:

After giving a name and a description to this action, SQL query language, should be used, to define which fields are to get from the messages, and in which situations the message should be grabbed:

```
select dev_id, metadata.time, payload_raw from '+/devices/+/up'
```

The ones being chosen are `dev_id`, `metadata.time` and `payload_raw` from all the messages sent uplink from all devices of all apps.

Then an action must be added, and afterward the option Split message into multiple columns of a DynamoDB table, is the one to choose:



Figure.4.43 – Send messages with DynamoDB as target

Then a new table, needs to be created, with the following properties:

Figure.4.44 – Creating a DynamoDB table for storing parking spaces messages

To finish the action creation, just add a role for the action to send data to DynamoDB:

The screenshot shows the 'Configure action' interface in the AWS IoT console. The title bar is blue with the text 'Configure action'. Below it, the action name 'Split message into multiple columns of a DynamoDB table (DynamoDBv2)' is displayed with a DynamoDB icon. A descriptive paragraph states: 'The DynamoDBv2 action allows you to write all or part of an MQTT message to a DynamoDB table. Each attribute in the payload is written to a separate column in the DynamoDB database. Messages processed by this action must be in the JSON format.' Below this, there is a section for 'Table name' with a dropdown menu showing 'parking_spaces' and a 'Create a new resource' button. Further down, a section titled 'Choose or create a role to grant AWS IoT access to perform this action.' contains a table with one row: 'aws-iot-dynamodb' with a green 'Policy Attached' status and a checkmark. To the right of this row are 'Create Role' and 'Select' buttons. At the bottom left is a 'Cancel' button, and at the bottom right is an 'Add action' button.

Figure.4.45 – Adding a role for AWS IoT action to write in DynamoDB

And to finish this step, just confirm the rule creation:

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

parking_space_uplink_messages

Description

action to fetch messages from Node (uplink) and persist information in DynamoDB

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

SELECT «Attribute» FROM «Topic Filter» WHERE «Condition». For example: SELECT temperature FROM /ot/topic WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```

1 select dev_id, metadata.time, payload.raw from "/devices/*/*/*up"

```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*) required

Split message into multiple columns of a DynamoDB table...
parking_spaces

Remove Edit >

Add action

Figure.4.46 – Finishing the rule creation

Consulting the DynamoDB table just created, the uplink messages being inserted, can be seen:

	dev_id	time	payload_raw
<input type="checkbox"/>	first_parking_spot	2019-09-19T14:30:59.327370753Z	AA==
<input type="checkbox"/>	first_parking_spot	2019-09-19T14:31:04.32624589Z	AA==
<input type="checkbox"/>	first_parking_spot	2019-09-19T14:31:09.329553123Z	AA==
<input type="checkbox"/>	first_parking_spot	2019-09-19T14:31:14.327845961Z	AA==
<input type="checkbox"/>	first_parking_spot	2019-09-19T14:31:19.334341791Z	AA==

Figure.4.47 – Uplink messages from the Node inserted at DynamoDB

Next step, now that the messages are already in the DynamoDB table, is to create a Lambda function that reads all this information and presents the parking space status in a web interface.

Create function

Choose one of the following options to create your function:

- Author from scratch** (Selected): Start with a simple hello world example.
- Use a Blueprint
- Import serverless app repository

Basic information

Function name:

Runtime:

Permissions: ☒ Create a new role with basic Lambda permissions

Execution role: ☒ Create a new role with basic Lambda permissions

Create function

Figure.4.48 – Lambda function creation

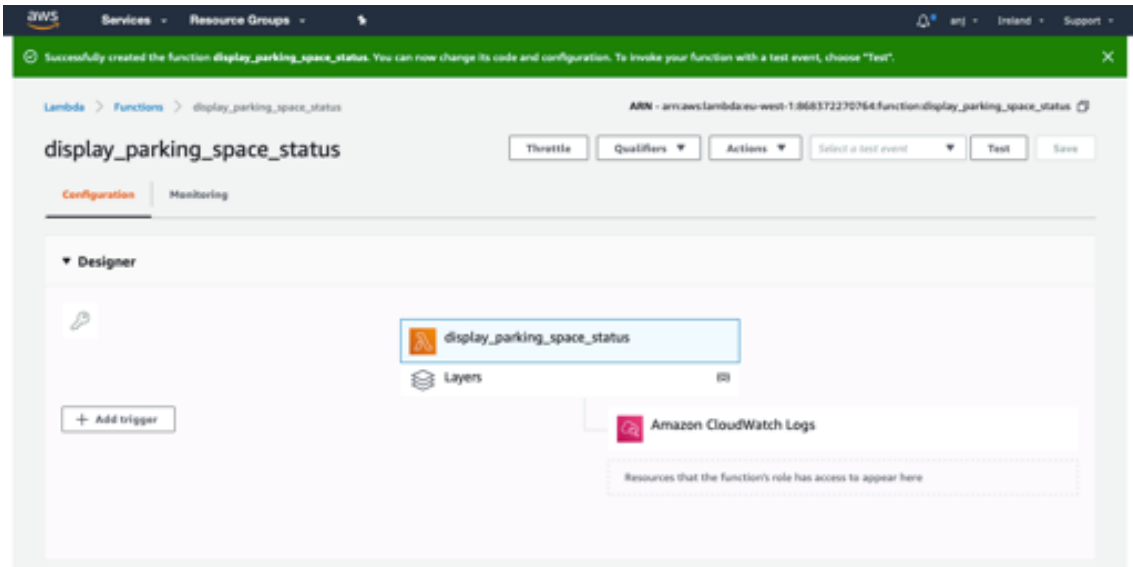


Figure.4.49 – Lambda function created and ready to be configured

One note: the role assigned to this Lambda function needs to be allowed to scan the DynamoDB table.

4.12 – Displaying the parking space status and allowing to reserve remotely using a simple interface

Assumption for this stage: the user that wants to make a reservation has previously registered in the system, had input its data, and his Bluetooth unique identifier is currently registered in the system. This can be accomplished by a mobile application to which the user allows access to the system information, mainly location and Bluetooth stack.

Now that the Lambda function is ready, it's time to develop the code, so that it reads the status of the parking space from the DynamoDB, so it can be shown to the user.

The code basically creates a handler, that is configured to scan the `parking_spaces` table, limited to 10 rows, and return this data, while logging the event to AWS CloudWatch.

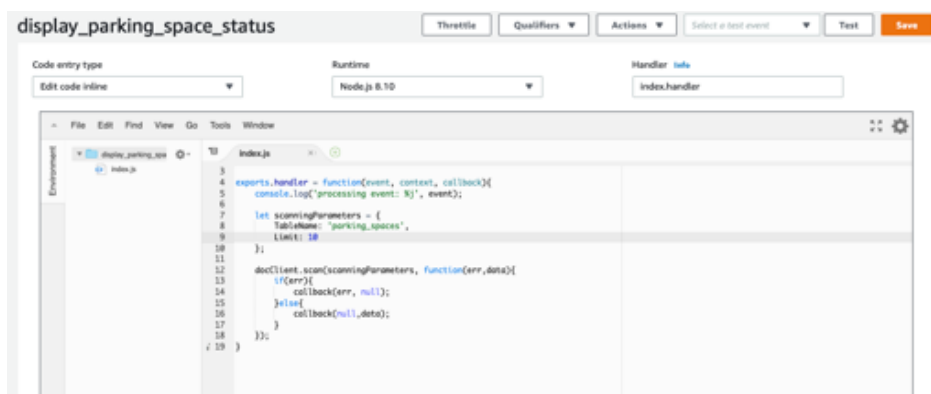


Figure.4.50 – AWS Lambda code Windows with node.js code for scanning the DynamoDB table

In this interface, the Lambda function can be tested, by clicking the Test button, and checking the results:



Figure.4.51 – AWS Lambda test execution results

Next step: to expose the Lambda execution to the user, it is needed to use the AWS API Gateway service[41] that will allow to expose the data in a secure way, through a REST API:

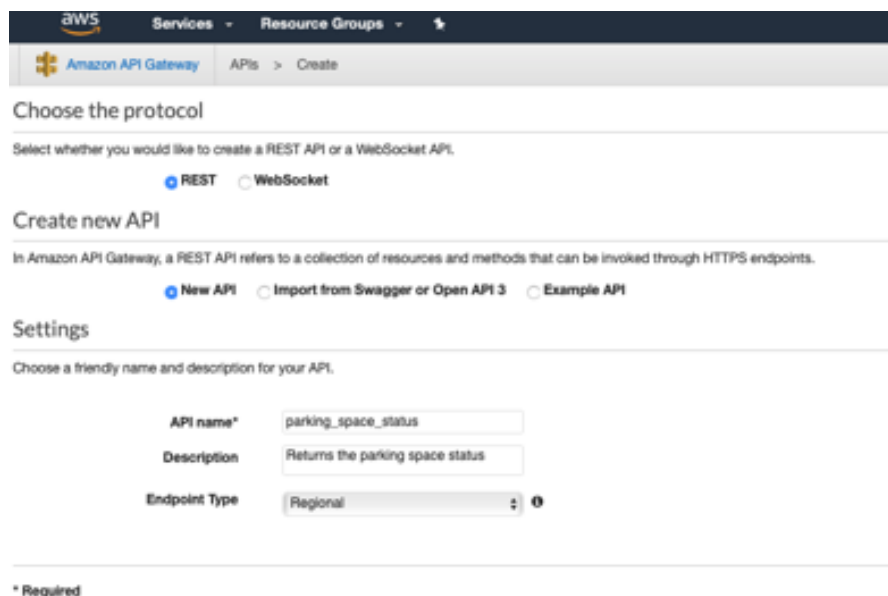


Figure.4.52 – Creating a new API in AWS API Gateway

Then it's time to define the method for the API – in this case the GET method, and associate it with the previously created Lambda function:

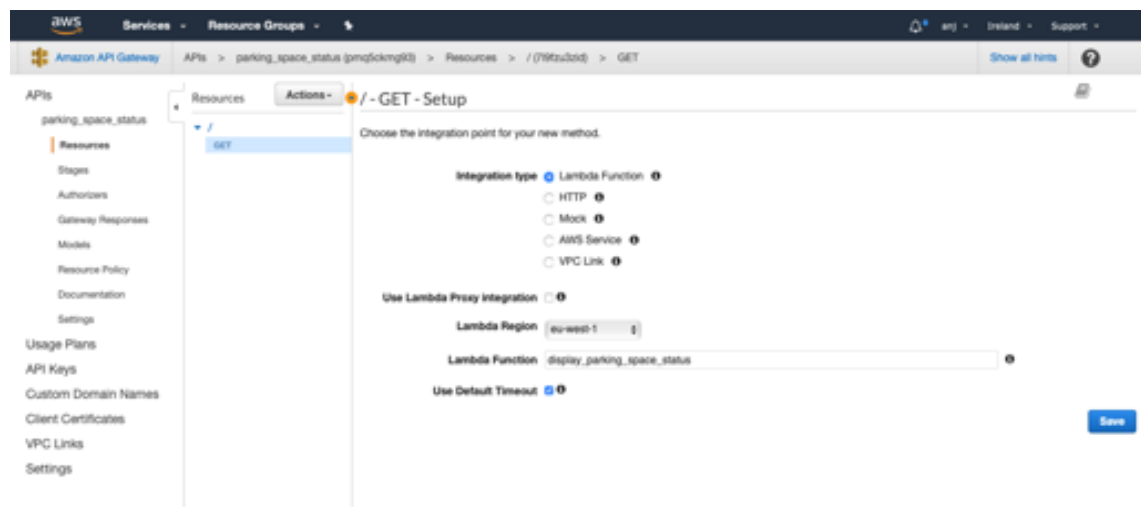


Figure.4.53 – Defining GET method and associating with the Lambda function

Afterwards this API is deployed in a available stage (in this case called beta):

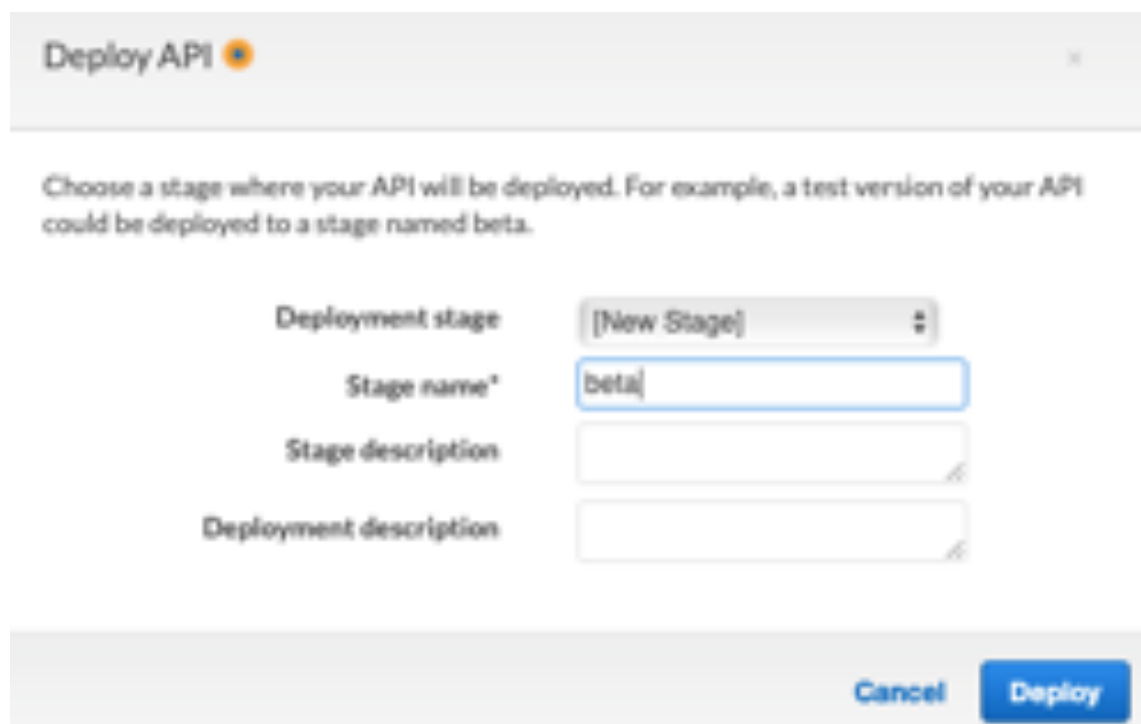


Figure.4.54 – Deploying the API in beta stage

Now, it is possible to access the API, by navigating to the beta stage URL, for the API Gateway, which is:

<https://pmq5ckmg93.execute-api.eu-west-1.amazonaws.com/beta>

Here the information from the Node, that is persisted in the DynamoDB, is available in a REST API format:

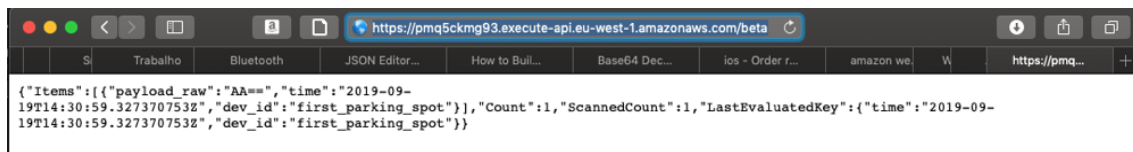


Figure.4.55 – REST API with parking space information

The last step for having the interface ready for the user, involves another AWS service, this time the AWS Simple Storage Service or S3[42]. This service stores data in object format, in a very scalable, performant and secure way.

The service provides what they call buckets, for the users to store objects in them. And this storage is used as the repository for the interface web pages.

The web pages are very simple, with no UI/UX work performed, and the main objective is just to display the parking space status. The pages are made in HTML, using a simple bootstrap template and make use of jQuery to consume the data from the REST API, and present it to the user.

First step is to configure the S3 bucket to receive the files. After the creation of the S3 bucket, named parkingspacestatus, its properties can be changed to make a static web-site repository:

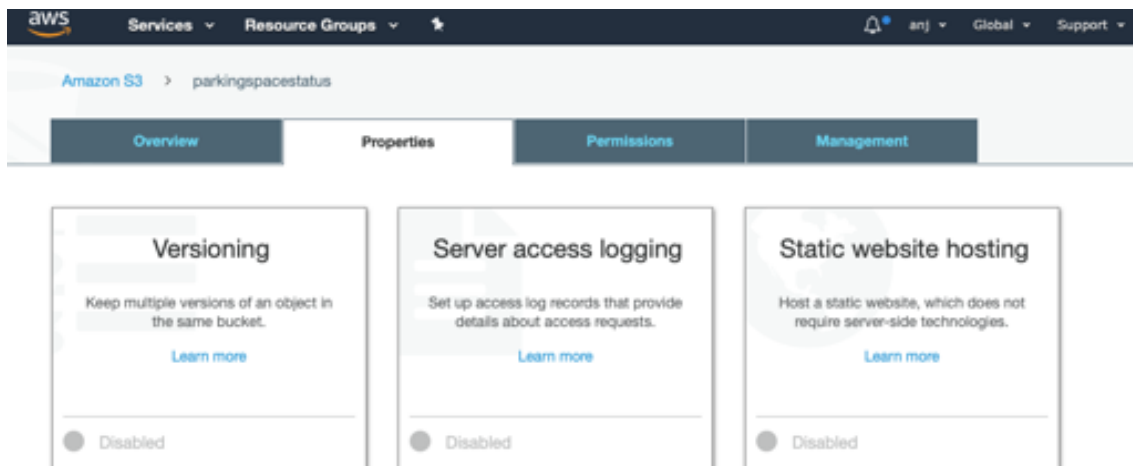


Figure.4.56 – S3 bucket properties

Static website hosting ✕

Endpoint : <http://parkingspacestatus.s3-website-eu-west-1.amazonaws.com>

☒ Use this bucket to host a website ⓘ [Learn more](#)

Index document ⓘ

Error document ⓘ

Redirection rules (optional) ⓘ

☐ Redirect requests ⓘ [Learn more](#)

☐ Disable website hosting

☒ Disabled

[Cancel](#) [Save](#)

Figure.4.57 – Enabling static website hosting

Now it is time to upload the files to the bucket, using the upload interface, and afterwards, the website is available for the final user, at the URL:

<http://parkingspacestatus.s3-website-eu-west-1.amazonaws.com/index.html>

Reserved Parking Validation - Parking Space Status		
This page shows status of the parking space for Master's Degree Dissertation "Reserved Parking Validation" by António Nunes Jorge		
Date	Parking Space Name	Status
2019-09-19 14:30:59	first_parking_spot	Available

Figure.4.58 – Web interface for Reserved Parking Validation project status

4.13 – Sending the reserved information to the Node and displaying the reservation status locally on the Node

Making use of the already established socket for communication between the Node and the Gateway, it was easy to send the reserved information to the Node.

On the Gateway side what was needed, was to send message with a known format: it was set that if the parking is reserved, the message sent is the MAC address of the user making the reservation. If what was needed was to cancel the reservation the payload to send to the Node would be 0x00 to signal the cancelation of the reservation.

```
64 #setting not reserved
65 reserved = b'\x00'
66
67 while True:
68     #receives
69     rx = s.recv(64)
70     if rx:
71         reserved = rx
72         print(rx)
```

Figure.4.59 – Node listening to messages from the Gateway

```
95 #state machine for parking
96 if reserved != b'\x00':
97     s.send(bytes([0x02])) # parking reserved
98     pycom.rgbled(0x0000FF)
99     while adv != reserved:
100         adv = bluetooth.get_adv()
101         if adv != None:
102             print("Found device with addr = {}".format(ubinascii.hexlify(adv.mac)))
103             time.sleep(1)
104     print("open")
```

Figure.4.60 – Checking the message payload and sending acknowledge

On the Node, when the payload is confirmed, the status LED is turned to blue, and an acknowledged message is sent to the Gateway, confirming the reserved state of the parking space.

4.14 – Scanning user proximity and allowing parking only to the user who made the reservation

Assumption for this stage: the user who made the reservation has a mobile device that is advertising its Bluetooth data.

When a user, comes near the parking space, the routine running on the Node, checks if the unique identifier (MAC address) being broadcast is the same, of the user who made the reservation. If they are the same, the Node issues a command to open the barrier, of the parking space.

The first step was to build code, that, running on the Node, would scan for Bluetooth advertisements, and would print the MAC address of the device.

```
1 from network import Bluetooth
2 import ubinascii
3 import time
4
5 bluetooth = Bluetooth()
6 bluetooth.start_scan(-1) # start scanning with no timeout
7
8 adv = None
9 while True:
10     adv = bluetooth.get_adv()
11     if adv != None:
12         print("Found device with addr = {}".format(ubinascii.hexlify(adv.mac)))
13         time.sleep(1)
14
```

Connected ✓ ble

Found device with addr = b'7073cbd9ffe9'

Found device with addr = b'7073cbd9ffe9'

Found device with addr = b'4f16fe638135'

Found device with addr = b'689ebefcbde0'

Found device with addr = b'792bb142afb0'

Found device with addr = b'689ebefcbde0'

Found device with addr = b'5c969d8aba2e'

Found device with addr = b'7073cbd9ffe9'

Found device with addr = b'4f16fe638135'

Found device with addr = b'7073cbd9ffe9'

Found device with addr = b'792bb142afb0'

Found device with addr = b'4f16fe638135'

Figure.4.61 – Scanning Bluetooth devices and showing MAC address for each advertisement

From here, it was just a matter of changing the main.py code, to introduce the BLE scan, and if the parking space, was reserved, and the user who reserved it, is within range, the Node issues an electric signal to the barrier system to open the parking space (in this code version still represented by a print(“open”).

```
95 #state machine for parking
96 if reserved != b'\x00':
97     s.send(bytes([0x02])) # parking reserved
98     pycom.rgbled(0x0000FF)
99     while adv != reserved:
100         adv = bluetooth.get_adv()
101         if adv != None:
102             print("Found device with addr = {}".format(ubinascii.hexlify(adv.mac)))
103             time.sleep(1)
104         print("open")
```

Figure.4.62 – Checking correct user presence and “opening” the barrier

The last change was to instantiate PIN 11, as an output:

```
from machine import Pin

# initialize 'P11' in gpio mode and make it an output
p_out = Pin('P11', mode=Pin.OUT)
```

Figure.4.63 – Setting PIN 11 to GPIO Mode and as an output

And when the user gets nearby the Node, PIN11 is set to 1 with `p_out.value(1)`.

4.15 – Road bumps, problems and questions

During the project execution and specially during this dissertation writing, there were some problems, that required pauses, reflection, deep reading and solution devisal.

One of them was, inevitably an occurrence of Murphy's Law[43] “Anything that can go wrong will go wrong”, was that, when running some tests for the dissertation elaboration, Atom IDE just decided to stop working with pymakr plugin:

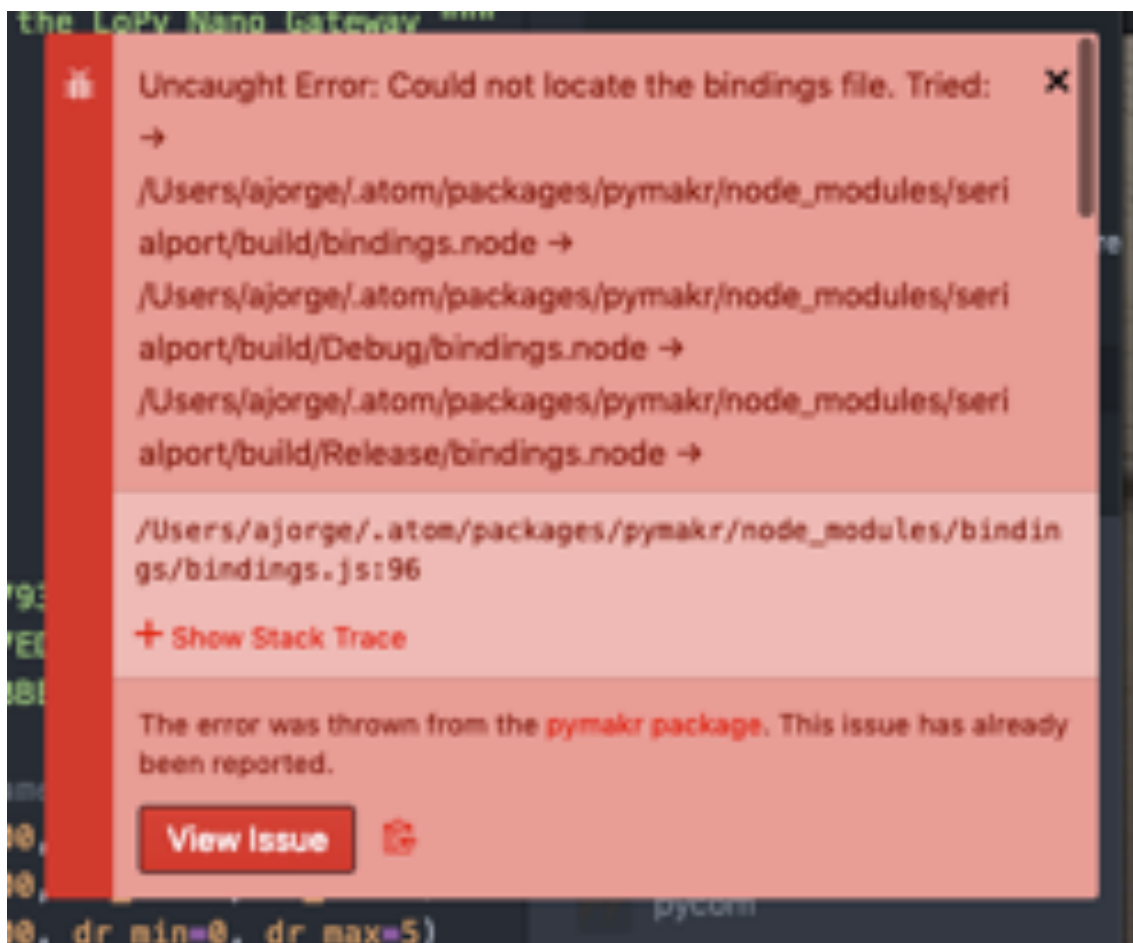


Figure.4.64 – Error on Atom using pymakr plugin

Making some search on the web, some incompatibility between the latest versions of Atom and the pymakr plugin were found.[44]

Commenters suggested to revert Atom two versions back. But in this case the problem wasn't solved.

Then another commenter, a developer of the pymakr plugin, suggested to overwrite a configuration file, on the pymakr plugin folder. But again, it did not solve this problem.

Last resort, to uninstall and reinstall both Atom and pymakr, which finally solved problem.

Other problem that arise during the dissertation writing, was that the Node would not join the Gateway network. That was a strange situation, as no change was made in the preceding weeks. All the keys and configurations were confirmed, and everything was correct.

Once again, resort to the world wide web was used, and after some research, someone with a similar problem[45] was found. The explanation was that after a while, and because of a bug, the application key (even if not expired or revoked) would not be accepted in the join procedure in OTAA. Cheap and dirty solution: revoke the key, generate a new one and update it in the Node software.

For testing purposes this would not case much problem, but in a production environment where hundreds or even thousands of devices may be provisioned, this solution would not due, and a further investigation and solution would need to be achieved.

5

Chapter 5 – Conclusions and Next Steps

It was a magnificent adventure to develop this project and be able to bridge the need for a solution with the several technologies available in the market.

During the several months that it took to build the project and make this dissertation, several solutions appeared in the market, and some of them are already commercial solutions, integrating sensors and communication technologies in one small, rugged, and easily installable package. And many of them choose similar technologies to the ones already chosen for the project.

This only reinforces the fact that the technologies chosen, were the correct ones, as they are spot on with several of these commercial solutions, that use LoRa as the communication layer, and magnetic sensors to understand the occupancy of the parking spot.

Accounting for the chosen methodology, it was a safe bet, mainly because, as being a worker-student, with reduced available schedule, the author could safely pause after a project phase, without worrying that, the author would have nothing to show up, if he had no more capability of driving another project phase.

Regarding the project next logical steps, to evolve from a PoC to a real project, would be the following:

- Prototype and implement a renewable energy harvesting system, that can reduce the already small footprint of energy consumption;
- Move from a fixed power only solution, to alternatives with battery usage and charging;
- Prototype, test and produce a container for putting the electronics parts (Node with magnetic sensor and parts);
- Prototype, test and produce a box to put all the signaling information inside (colored LEDs, etc.);

- Develop and integrate the solution with a commercial LoRa gateway;
- Develop and test a mobile app application that integrates with the solution and provides a usable mobile solution for end-users. This would require UI/UX tasks that were out of scope for this dissertation, and also the mobile development that would by itself fulfill another master's degree dissertation requirements. Also, all the registration and authorization workflow need to be developed and integrated in the app;
- Make a pitch proposal to EMEL (Empresa Municipal de Estacionamento de Lisboa) which is responsible for managing the parking in the city of Lisbon, having them install this solution in the city of Lisbon's disable parking spaces, integrating the app and system with their georeferencing system.

It was a very special project, even knowing that the solution might never see the light of day as a finished/ready to market product, but at least knowing that this work could serve as inspiration or as starting point, for someone, to move the idea/concept into a product, or use some parts of it for another project, it's very rewarding.

Even the fact that to build this project, the need to study the reality of disabled people, understanding the asymmetries and obstacles, that every day, these people face, while trying to live their lives, as regular as possible, and understanding that any little thing this project, could accomplish, even if it was only raising awareness for these situations, was already a big sense of mission accomplished.

Bibliography

- [1] P. White, "No Vacancy - Park Slope's Parking Problem And How to Fix It," *Transalt.org*, p. 22, 2007.
- [2] C. Harrison *et al.*, "Foundations for Smarter Cities," *IBM Journal of Research and Development*, vol. 54, no. 4, pp. 1–16, Jul. 2010.
- [3] H. Arasteh *et al.*, "Iot-based smart cities: A survey," in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, 2016, pp. 1–6.
- [4] J. Gu, Z. Zhang, F. Yu, and Q. Liu, "Design and implementation of a street parking system using wireless sensor networks," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 1212–1217.
- [5] J. Yang, J. Portilla, and T. Riesgo, "Smart parking service based on Wireless Sensor Networks," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 6029–6034.
- [6] Yusnita Rahayu and Fariza N. Mustapa, "A Secure Parking Reservation System Using GSM Technology," *International Journal of Computer and Communication Engineering*, pp. 518–520, 2013.
- [7] P. DharmaReddy, A. RajeshwarRao, and D. S. M. Ahmed, "An Intelligent Parking Guidance and Information System by using image processing technique," vol. 2, no. 10, p. 5, 2013.
- [8] L. Lambrinos and A. Dosis, "DisAssist: An internet of things and mobile communications platform for disabled parking space management," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, 2013, pp. 2810–2815.
- [9] M. Patil and V. N. Bhonge, "Wireless Sensor Network and RFID for Smart Parking System," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 4, Apr. 2013.
- [10] H. Singh, C. Anand, V. Kumar, and A. Sharma, "Automated Parking System with Bluetooth Access," 1, vol. 3, no. 05, 2014.

- [11] H. Al-Kharusi and I. Al-Bahadly, "Intelligent Parking Management System Based on Image Processing," *World Journal of Engineering and Technology*, vol. 02, no. 02, pp. 55–67, 2014.
- [12] H. Chien Yee and Y. Rahayu, "Monitoring Parking Space Availability via Zigbee Technology," *International Journal of Future Computer and Communication*, vol. 3, no. 6, pp. 377–380, Dec. 2014.
- [13] Z. Suryady, G. R. Sinniah, S. Haseeb, M. T. Siddique, and M. F. M. Ezani, "Rapid development of smart parking system with cloud-based platforms," in *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, 2014, pp. 1–6.
- [14] O. Orrie, B. Silva, and G. P. Hancke, "A wireless smart parking system," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 004110–004114.
- [15] E. Karbab, D. Djenouri, S. Boulkaboul, and A. Bagula, "Car park management with networked wireless sensors and active RFID," in *2015 IEEE International Conference on Electro/Information Technology (EIT)*, 2015, pp. 373–378.
- [16] R. Salpietro, L. Bedogni, M. D. Felice, and L. Bononi, "Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 18–23.
- [17] Baratam. M Kumar Gandhi, "A Prototype for IoT based Car Parking Management system for Smart cities," *Indian Journal of Science and Technology*, vol. 9, no. 17, May 2016.
- [18] M. Amin, R. Kawaguchi, N. Shirmohammad, and M. Sato, "BlueParking: An IoT based Parking Reservation Service for Smart Cities," in *Proceedings of the Second International Conference on IoT in Urban Space - Urb-IoT '16*, Tokyo, Japan, 2016, pp. 86–88.
- [19] E. Muñoz, "D1.1 Project Handbook - Simon - Assisted Mobility for Older and Impaired Users - Public Deliverables," p. 36, Aug. 2014.
- [20] E. Muñoz, "D3.1 Reference Architecture and Principles - Simon - Assisted Mobility for Older and Impaired Users - Public Deliverables," p. 36, Jan. 2014.
- [21] "phonepark." [Online]. Available: <http://www.phonepark.pt/index.php/pt/home>. [Accessed: 12-Feb-2019].
- [22] N. Ahmed, H. Rahman, and Md. I. Hussain, "A comparison of 802.11ah and 802.15.4 for IoT," *ICT Express*, vol. 2, no. 3, pp. 100–102, Sep. 2016.
- [23] ELIFTECH, "7 Reasons Why AWS IoT Platform Is Great for IoT Startups," *IoT For All*, 26-Jul-2018. .
- [24] Espressif Systems, "ESP32 Datasheet." [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: 15-Feb-2019].
- [25] Altice Labs, "IoT Cellular Networks," Oct. 2017.
- [26] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, "A survey on IEEE 802.11ah: An enabling networking technology for smart cities," *Computer Communications*, vol. 58, pp. 53–69, Mar. 2015.

- [27] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Express*, vol. 3, no. 1, pp. 14–21, Mar. 2017.
- [28] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Express*, Jan. 2018.
- [29] GSM Association, "The Mobile Economy Europe 2018," 2018.
- [30] PyCom, "PyCom LoPy 4 Datasheet." [Online]. Available: https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf. [Accessed: 15-Jun-2019].
- [31] A. Industries, "Adafruit Feather 32u4 RFM96 LoRa Radio - 433MHz." [Online]. Available: <https://www.adafruit.com/product/3079>. [Accessed: 24-Aug-2019].
- [32] "LoRaWAN Nano-Gateway." [Online]. Available: <https://docs.pycom.io/tutorials/lora/lorawan-nano-gateway/>. [Accessed: 16-Sep-2019].
- [33] "Atom." [Online]. Available: <https://docs.pycom.io/pymakr/installation/atom/>. [Accessed: 17-Sep-2019].
- [34] "LoPy," *The Things Network*, 10-Sep-2019. [Online]. Available: <https://www.thethingsnetwork.org/docs/devices/lopy/>. [Accessed: 16-Sep-2019].
- [35] Honeywell, "HMC5883L 3-Axis Digital Compass IC." [Online]. Available: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf. [Accessed: 20-Feb-2019].
- [36] "General Data Protection Regulation (GDPR) Compliance Guidelines," *GDPR.eu*. [Online]. Available: <https://gdpr.eu/>. [Accessed: 18-Sep-2019].
- [37] "What Is AWS Elastic Beanstalk? - AWS Elastic Beanstalk." [Online]. Available: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html>. [Accessed: 18-Sep-2019].
- [38] "AWS CloudFormation - Infrastructure as Code & AWS Resource Provisioning," *Amazon Web Services, Inc.* [Online]. Available: <https://aws.amazon.com/cloudformation/>. [Accessed: 18-Sep-2019].
- [39] "Amazon DynamoDB - Overview." [Online]. Available: <https://aws.amazon.com/dynamodb/>. [Accessed: 19-Sep-2019].
- [40] "What Is AWS Lambda? - AWS Lambda." [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>. [Accessed: 19-Sep-2019].
- [41] "Amazon API Gateway," *Amazon Web Services, Inc.* [Online]. Available: <https://aws.amazon.com/api-gateway/>. [Accessed: 19-Sep-2019].
- [42] "Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service," *Amazon Web Services, Inc.* [Online]. Available: <https://aws.amazon.com/s3/>. [Accessed: 19-Sep-2019].
- [43] "Murphy's law," *Wikipedia*. 29-Aug-2019.
- [44] "Failed to load the pymakr package · Issue #127 · pycom/pymakr-atom," *GitHub*. [Online]. Available: <https://github.com/pycom/pymakr-atom/issues/127>. [Accessed: 18-Sep-2019].

- [45] “No Join Response from TTN,” *The Things Network*, 06-Nov-2018. [Online]. Available: <https://www.thethingsnetwork.org/forum/t/no-join-response-from-ttn/10469/24>. [Accessed: 18-Sep-2019].